

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-285154

(43)Date of publication of application : 23.10.1998

(51)Int.Cl. H04L 9/26
G09C 1/00
G09C 1/00
H04L 9/08

(21)Application number : 09-090873

(71)Applicant : METEOOLA SYST KK
AIRU:KK

(22)Date of filing : 09.04.1997

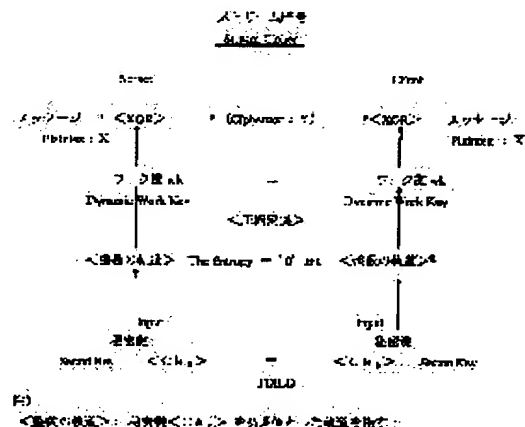
(72)Inventor : WATANABE EIJI
SEKIGUCHI YOSHIHIRO

(54) KEY GENERATION SYSTEM CONSTITUTING PERFECT SECRECY CIPHER SYSTEM, KEY SHARED PROTOCOL WITH AUTHENTICATION, 'ONE-TIME STREAM CIPHER', 'ONE-TIME PASSWORD' AND KEY MANAGEMENT ALGORITHM

(57)Abstract:

PROBLEM TO BE SOLVED: To give informational logical security to an open network.

SOLUTION: In a key generation system which provides both terminals that perform communication in a client/server network, etc., with a register which operates with $2u$ ($u \geq 4$) adic number, and is provided with a mapping algorithm part that executes map of an integer space of the register ($I_s \rightarrow I_s$) to its own and generates an 'integer orbit' and a compressing and converting part which compresses and converts the 'integer orbit' and creates a binary 'key stream', when inverse map that traces back a map algorithm to the past is attempted, not only 'n-bits entropy' occurs, but also 'informational logical security map algorithm' to which 'entropy' that accompanies compressive conversion is also added is mounted to n-times inverse map and is 'intentionally' embedded in the algorithm of cipher and decoding.



LEGAL STATUS

[Date of request for examination] 09.04.2004

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision
of rejection]

[Date of requesting appeal against examiner's
decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平10-285154

(43) 公開日 平成10年(1998)10月23日

(51) Int.Cl. ⁸	識別記号	F I
H 0 4 L 9/26		H 0 4 L 9/00 6 5 9
G 0 9 C 1/00	6 1 0	G 0 9 C 1/00 6 1 0 D
	6 5 0	6 5 0 B
H 0 4 L 9/08		H 0 4 L 9/00 6 0 1 E

審査請求 未請求 請求項の数 6 O L (全 25 頁)

(21) 出願番号 特願平9-90873

(22) 出願日 平成9年(1997)4月9日

(71) 出願人 591223231

メテオーラ・システム株式会社

神奈川県横浜市港北区高田町1549番地

(71) 出願人 596059163

株式会社アイル

東京都中野区本町2丁目50番8号

(72) 発明者 渡邊 榮治

神奈川県横浜市港北区高田町1549番地

(72) 発明者 関口 義浩

神奈川県伊勢原市石田200 メテオーラ・システム株式会社内

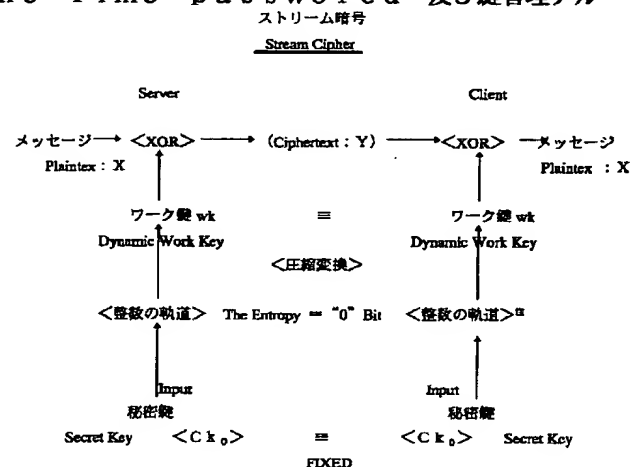
(74) 代理人 弁理士 三好 秀和 (外 8 名)

(54) 【発明の名称】 完全守秘性暗号系を構成する鍵生成システム、認証付き鍵共有プロトコル、“One-Time Stream Cipher”、“One-Time password”及び鍵管理アル

(57) 【要約】 (修正有)

【課題】 オープンネットワークに情報理論的な安全性 “Unconditional Security” を与える。

【解決手段】 クライアント／サーバー・ネットワーク等で通信を行う端末双方に、 2^u ($u > 4$) 進数で演算するレジスタを備え、レジスタの整数空間 I_s のそれ自身への写像 ($I_s \rightarrow I_s$) を実行して「整数の軌道」を生成する写像アルゴリズム部、及び「整数の軌道」を圧縮変換して2進の「鍵ストリーム」を生成する圧縮変換部を備えた鍵生成システムにおいて、写像アルゴリズムを過去に遡る逆写像を試みると、 n 回の逆写像に対しては“ n ビットのEntropy”が生じだけでなく、圧縮変換に伴う“Entropy”もそれに加わる「情報理論的に安全な写像アルゴリズム」が実装され、暗号と復号のアルゴリズムの中に「意図的に」埋め込む。



注)
＜整数の軌道＞：秘密鍵 Ck_0 を初期値とした軌道を指す；

【特許請求の範囲】

【請求項1】 クライアント／サーバー・ネットワーク（以下C/Sネットワーク）などで通信を行う端末双方に、 $2u$ ($u > 4$) 進数で演算するレジスターを備え、該レジスターの整数空間 I_s のそれ自身への写像 ($I_s \rightarrow I_s$) を実行して「整数の軌道」を生成する写像アルゴリズム部と、該 $2u$ 進数の「整数の軌道」を圧縮変換して2進の「鍵ストリーム」を生成する圧縮変換部、とを備えた鍵生成システムを実装し、該鍵生成システムには、その写像アルゴリズムを過去に遡る逆写像を試みると、 n 回の逆写像に対しては“ n ビットのEntropy”が生じだけでなく、上記圧縮変換に伴う“Entropy”もそれに加わる「情報理論的に安全な写像アルゴリズム」が実装されており、該情報理論的に安全な写像アルゴリズムを「鍵の情報理論的な安全性」を追求する目的で、暗号と復号のアルゴリズムの中に「意図的に」埋め込んだ事の特徴とする情報理論的に安全な対称暗号系。

【請求項2】 前記対称暗号系には、該写像アルゴリズムによる整数軌道の任意の軌道値“ ck_0 ”を「秘密鍵」として共有すると同時に、実際に暗号／復号するワーク鍵には上記「鍵ストリーム」を使う一方、秘密鍵 $<ck_0>$ がどちらかの該鍵生成システムにインプットされると、該秘密鍵を情報理論的に安全な秘密鍵にするために、該写像アルゴリズムを規定回数 (i 回) 実行させて、該秘密鍵 $<ck_0>$ の整数軌道 ($ck_1, ck_2, ck_3, \dots, ck_i$) を作らせると共に、該整数軌道値を該端末内に「保存」という規約と、さらに、この保存されている i 番目の整数軌道値 $<ck_i>$ を初期値とした新たな整数軌道 ($ck_i, ck_{i+1}, \dots, ck_{i+j}$) を生成させて、この後者の整数軌道を圧縮変換した「鍵ストリーム」をワーク鍵に使うという規約と、これら二つの規約から構成される「安全な同期プロトコル: Secure Synchronous Protocol: SSP」を実装し、該「安全な同期プロトコル」を共有する事により、対称暗号系の共通鍵の安全性を計算量の安全性“Computational Security”ではなく、情報理論的に安全な、つまり“Unconditional Security”の問題に帰着させたことを特徴とする対称暗号系。

【請求項3】 秘密鍵 $<ck_0>$ を共有している前記

拡張ワーク鍵の一致 = 拡張秘密鍵を共有する、

【数4】

$$\begin{aligned} &= <ck_i.serv> = <ck_i.clin> ; i = 1 \sim \text{任意} \\ &= <ck_0.serv> = <ck_0.clin> ; \\ &= <ck_0> ; \text{秘密鍵を共有する、} \end{aligned}$$

を無条件に満足する共通鍵群となるので、

“login”が発生する度に、端末双方とも一回使用した

1、2項記載の対称暗号系は、互いに相手を確認しうる手段 (ID) として共有している秘密鍵 $<ck_0>$ を、認証の手段としてネットワーク上で交換せず、保持すると共に、

対称暗号系のサーバー上で他の偶然的イベントを初期値とした別の整数の軌道値を生成し、これを前記安全な同期プロトコル (SSP) の下でサーバーが保持しているクライアントの秘密鍵 $<ck_0.serv>$ によって暗号化し、該秘密鍵 $<ck_0.serv>$ の「代理の鍵 $<sk_i.serv>$ 」として他方のクライアントに配送する一方、この暗号データを受け取ったクライアントも、該クライアントが保持していた秘密鍵 $<ck_0.clin>$ により同じく前記“SSP”の下で該暗号データを復号化して「代理鍵」 $<sk_i.clin>$ を受け取るという規約「鍵の同期配送プロトコル」を実装し、

さらに、サーバーは代理鍵 $<sk_n.serv>$ を初期値とした整数軌道を生成して該拡張代理鍵群を保持し；

【数1】 $\{sk(n+1).serv, sk(n+2).serv, sk(n+3).serv, \dots\}$ 、

一方クライアントも代理鍵 $<sk_n.clin>$ について、それを初期値とした 整数軌道を生成して該拡張代理鍵群を保持し；

【数2】 $\{sk(n+1).clin, sk(n+2).clin, sk(n+3).clin, \dots\}$ 、

これら拡張代理鍵群をそれぞれ前記鍵生成システムの圧縮変換部に入力し、それぞれ対応する拡張ワーク鍵系列を生成し、かつ保持し；

【数3】 $\{, wk_1.serv, wk_2.serv, wk_3.serv, \dots\}$

$\{, wk_1.clin, wk_2.clin, wk_3.clin, \dots\}$

これら情報理論的に安全な拡張ワーク鍵群をネットワークへ出して交換するという規約「鍵の同期認証プロトコル」を実装し、

該「鍵の同期配送プロトコル」と「鍵の同期認証プロトコル」の両者から構成される「認証付きの鍵共有プロトコル: Authenticated Key Agreement Protocol」を実装する事の特徴として秘密鍵の共有を実現する前記1、2項記載の対称暗号系。

【請求項4】 対称暗号系の端末双方が“login”後、前記1、2項記載の安全な同期プロトコル (SSP) の下で前記3項記載の「認証付き鍵共有」を成した後に於いては、当該対称暗号系の中の鍵群は、

秘密鍵を該共通鍵群の中の未使用の拡張秘密鍵をもって「更新する」ことを特徴とした対称暗号系又は、前記

1、2、3項記載の構成要件を満たす“One-Time Stream Cipher”、同じく前記1、2、3項記載の構成要件を満たすことを特徴とした“One-Time Password”。

【請求項5】 多数のクライアントの秘密鍵のファイル「鍵箱」を備えたサーバーが、その鍵箱を情報理論的に安全なセキュリティで守るために、前記2項記載の安全な同期プロトコル(SSP)の下で前記3項の「認証付き鍵共有」を成した後に於いて、

該サーバーは、「鍵箱」の任意の秘密鍵 $\langle ck_0 \rangle$ を前記1項記載の当該秘密鍵 $\langle ck_0 \rangle$ の「鍵ストリーム」 $\langle wk - ck_0 \rangle$ によって該鍵 $\langle ck_0 \rangle$ 自身を暗号化し、該「鍵箱」を暗号ファイルCRYPTO($ck_n : n=0, 1, 2, 3, \dots$)として管理する一方、

前記1乃至4項記載の対称暗号系がこれを復号化するには、該クライアント自身が保持している鍵 $\langle ck_0.clin \rangle$ から生成した前記1、2項記載の鍵ストリーム $\langle wk - ck_0.clin \rangle$ しか復号化の手段が残されていないので、

前記1乃至4項記載の対称暗号系は、次の“login”時に、該クライアントの該鍵ストリームをサーバーに配送し、

該サーバー上で暗号ファイル CRYPTO($ck_n : n=0$)を復号化し、当該クライアントの秘密鍵 $\langle ck_0.serv \rangle$ をサーバーへ与えるとした「鍵管理アルゴリズム」を備え、

これにより該サーバーは該秘密鍵 $\langle ck_0.serv \rangle$ を入手した後は、前記3項記載の規約「認証付き鍵共有プロトコル」の実行に入ることを特徴とした鍵管理アルゴリズム及び鍵サーバー。

【請求項6】 前記1項記載の鍵生成システムのレジスターは、 10^n ($n \geq 1$) 個または 2^n 個 ($n > 1$) の桁から成立する一様レジスターであり、その一桁も又 10^m 個または 2^m 個の桁から成立する ($10^n \leq 10^m$) 一様レジスターであり、というように「入れ籠構造」を成した「自己相似形レジスター」であることを特徴とする一方、

該鍵生成システムの写像アルゴリズムは、該一様レジスターを 10^n 進数または 2^n 進数の整数空間 I_s と見なす該整数空間のそれ自身への写像演算 F である；($F : I_s \rightarrow I_s$)。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、従来のネットワーク・セキュリティが計算量的安全性“Computational Security”に依存していたのに対して、オープンネットワークに情報理論的な安全性“Unconditional Security”を与える実装技術に関する。

【0002】① 情報理論的な安全な鍵生成システム

② 情報理論的な安全な認証技術

③ 情報理論的な安全な鍵配送技術

④ 情報理論的な安全な鍵保管技術

⑤ 情報理論的な安全なメッセージの暗号技術

⑥ 情報理論的な安全なデジタル署名技術

に亘る。

【0003】

【従来の技術】インターネットにしろイントラネットにしろ、そこで必要とされるセキュリティは、上述6つの分野に亘る。これらセキュリティを実現する暗号技術には対称暗号方式と非対称暗号方式とがあり、現在この二方式が合い補ったハイブリッド方式がネットワーク・セキュリティを達成したかに見えている。対称暗号方式は、暗号鍵と復号鍵とが同じ対称鍵である。メッセージの機密保持を確保するためには、この対称鍵を秘密にしなければならないので、これを「秘密鍵」と呼んでいる。

【0004】歴史上ネットワーク上で、この秘密鍵をどのようにして共有するかを解決したのが、Diffie-Hellmanの鍵配送方式である。又、ここからRSAも工夫され、非対称暗号そのものが確立したことはよく知られている。

【0005】現在、上述6つの技術分野は、⑤のメッセージの暗号に対称暗号を適用し、その他の分野は非対称暗号の応用技術を適用するというハイブリッド方式によってカバーされている。この内、②の個人認証技術に関しては、実用上もっと手軽な技術が普及しつつある。

【0006】例えば、“challenge & Response”という手法を適用して、クライアントがサーバーにアクセスすると、毎回サーバーは“challenge number”という平文を送り出し、これを受け取ったクライアントは“password”という暗号文をサーバーに返す。これがResponseである。クライアントとサーバー双方にはDESなどの対称暗号ソフトがあらかじめインストールされていることは言うまでもない。

【0007】毎回平文と暗号文とがネットワーク上に送出されているので、このセキュリティを破ることは国家機関にとっては容易である。一方、民間ではこれで十分のセキュリティを達成しているようである。もっと信頼の於ける認証技術としては、One-Time password (OTP; RFCにてOTPと称されている)がある。難点は、使用回数が限られているので、商取引には向かない。RSAが現在主流の認証技術である。しかし、これは計算量的安全性に依存していることを忘れてはならない。

【0008】

【発明が解決しようとする課題】従来知られている暗号は“計算量的に安全である”と主張している技術である；これは、「Computational Security」と言われている。この種の技術が、現在オープンネットワークのスタンダードになっている。例えば、DESやRSAが有名である。

【0009】それはルーチンワークの大好きなコンピュ

ータで破ろうとする限り、安全であることが保証されている暗号である。しかし、RSAに関して言えば、将来素数分布に関する新しい数学が生まれたらどうなるであろうか（リーマン予想）。整数の素因数分解を容易に行えるようになるであろうし、その時、世界中を飛び回っている電子マネーは、電子マネーでなくなる日となる。“計算量的に安全である”とは、このような意味での相対的安全性であって、将来に亘って安全であるという意味ではない。

【0010】もう一つ困ったことは、従来の暗号技術は国家の安全保証とリンクしていることである。従って、これら暗号技術の民間ユースの転用に対して色々と制限が入ってくるのは当然である。

【0011】故に、オープンネットワークで使える新しい暗号技術の出現が望まれる。この世界でスタンダードになるセキュリティを実現できる暗号技術は、どんな条件に耐えねばならないであろうか；その技術が全ての人に公開され、評価され、誰でもそのソフトウェアが入手可能になるという事態を、まず受入なければならない。

【0012】かつてのように一部の人間だけが技術を共有し、つまり技術そのものを非公開の扱いにすることに依ってセキュリティを保つという秘密主義は、オープンネットワークに通用しない。従って、単に暗号システムを複雑にすることによっては、問題の解決にならないことが判る。

【0013】Gilbert Vernumによって1917年に発表されたというOne-Time Padは、単純にして、かつ“完全守秘性”注¹を提供する最初の実用的な技術であることが知られている。しかし、この技術は上述のスタンダードに耐えうる条件を備えているにもかかわらず、オープンネットワークに適用できない深刻な問題を内包している。そのアポリアとは、「鍵の配送と鍵の管理」に関することである；これは、鍵を配送するのに当該通信路以外に、別に安全な通信路を必要とするというアポリアである。

【0014】従って、非公開の扱いを可能とし、鍵の配送を特別な人が行う軍事、外交上の通信のみに使われているということである。完全守秘性の概念は、この暗号の実用化に後れて30年後に、C.E.Shannon（注1）によって与えられたという。この意義は、今日オープンネットワークの時代に特に大きい。何故なら、従来のオープンネットワークに於ける暗号技術が「Computational Security」という相対的安全性に依存しているのに対して、「確率的な答え」しか許さない暗号系の理論的存在とその安全性を提示したことである。判りやすく言えば、その暗号技術を全て公開し、裏も表もなく公開し、あらゆる技術進歩を計算に入れても、その暗号を“破れる”のは、唯一確率論しか無いと言う暗号の安全性を明らかにしたことである。このような暗号系は「情報理論的に安全である」とか、「Unconditional Security」

ty」であると言われている。俗な言い方をすれば、ハッカーは、サイコロの目が 2^{256} もあるようなサイコロを振る予想しか出せないし、コンピュータのプログラムは暗号文を解析して平文を作ったとしても、あらゆる平文が等確率に現れるから、やはりサイコロを振る結果になる。

【0015】結論を言えば、オープンネットワークでスタンダードになれるセキュリティ技術とは、こういう“Unconditionally Secure”な暗号系に依存すべきである。

【0016】しかし、この種の安全な暗号系、例えば“One-Time Pad cipher”をネットワークへ適用しようとする（注3）、「鍵の配送と鍵の管理」を他の暗号系または他の安全な通信路に依存しなければならないという深刻な問題に出会うので、このアポリアをどのように始末するか、ここが新しい暗号理論とその実装手段の試金石になるところであるし、本発明がブレイクスルーした点でもある。

【0017】

【課題を解決するための手段】暗号技術の基本は鍵生成システムにある、言い換えると、暗号技術の基本は乱数生成のアルゴリズムに依存する。従来の乱数生成アルゴリズムは“Computationally Secure”であっても、決して“Unconditionally Secure”な乱数生成法ではない、あるいは情報理論的な安全性の側面から研究された痕跡がない。

【0018】この“Unconditionally Secure”な乱数生成理論の一つが、「カオス発生の原因であるフラクタル構造そのもの」の中に発見された。カントール集合は「情報理論的に安全な」アルゴリズムであるという発見である。単に、カオスの現象作り出してそれから乱数列を取り出すという意味ではない。言い換えれば、コンピュータでカオス的乱数を作り、乱数列を生成したと言う人が居ても、それが「情報理論的に安全な」乱数生成手段であるという主張にはならないし、また情報理論的に安全な乱数の利用をしているという主張にもならない。さらにカオス理論で作った乱数なら安全であるとか、逆に簡単な非線型関数を使っているから安全ではないなどの議論も、この“Unconditional Security”に関する議論と全く関係ない。この点は特に強調しておかねばならない。

【0019】1) カオス写像の情報理論的に安全な一方変換性；どのような非線形曲線を使うにしろ、カントール集合を成す数学上のカオス写像によって生成されたカオス軌道は（Chaos Orbit）、未来に向かっては一意に決定される一方、過去に向かうと（ n 回逆写像しようと試みると） 2^n の不確定度を生じる（カントール集合を逆にたどる）。この n 回逆写像の“Entropy”を考えてみよう。

【0020】カオス軌道は互いに交わらない、つまり同

じ数学上の点を通らないという性質があるので、有限の時間内で「カオス軌道上の点は全て等確率で出現する」と言える。このカオス写像の性質を情報理論の視点で見ると、「 n 回逆写像のEntropyは n ビットである」ことが判る（等確率で出現するから）。ここで注意すべきは、数学上のカオス軌道と実装技術上のカオス軌道とは一致しない；何故なら実装技術上のカオス軌道は有限精度の整数の軌道であるから、単にカオス理論をプログラムしてもカオスにならないのである。そこで話を数学上のカオスに限るなら「過去のカオス軌道値は、現在の軌道値より n ビット余計な不確定度を帯びる、つまり、 n ビットの情報量を担う」という情報理論的な構造を発見する。（この情報理論的な構造は必ずしもカオス写像のみに発見されるとは限らないが、カオス理論はすでに確立しているので安心して使える。故に、以下カオス写像を例にして「Unconditionally Secure」な鍵生成理論を解説する。）

この情報理論的な構造に着目すると、我々は任意の瞬間のカオス軌道値そのものが「情報理論的に安全な秘密鍵」として使えるというアイデアを発見する。何故なら暗号を破るという行為は、「Recipient」側にある現在のデータから、「過去に遡って」「Originator」側の原因を特定する作業であるから、 n 回逆写像の「Entropy」が n ビットであるとは、「情報理論的に安全な鍵」を提供するアルゴリズムになる。一方この秘密鍵は、そのカオス軌道を「未来に向かっては一意に決定する」から「復号鍵：unique—decipherability」として機能させる事も出来る。これは暗号学的には、真に出来すぎた属性である。ここに改めて、我々は「復号鍵」として機能するカオス軌道上の任意の瞬間のカオス軌道値を「情報理論的に安全な拡張秘密鍵：Secret Key-Enhanced」と名付けると共に、該拡張秘密鍵から見て、 n ビット余計な不確定度を帯びている過去のカオス軌道値を「秘密鍵」と定義する。これらのカオス軌道値を、アイデア上の「カオス鍵」と言う場合がある。（詳細は図—2にて説明する）。上記のように、数学上のカオス写像をC.E. Shannon.の情報理論注1から再検討すると、カオス軌道は「情報理論的に安全な一方向変換性を備えている」ことが判る。この情報理論的発見から、我々は完全守秘性暗号系「Perfect SecrecyCryptosystem」に関する基礎理論を確立した。このカオス写像自体に関しては、既に思想として、1990年特願平2-226500「無線式電話の秘話システム及び秘話通信用LSI」の中で「秘話OS」として提示し説明しているところである。

【0021】今回、視点をShannoの研究に移し、コントロール集合を成すアルゴリズムを情報理論的な安全性を実現する基礎理論として採用した。

【0022】実装技術上のカオス写像を「カオスの写像」注2と表現する。単にカオス写像という時は数学上の「情報理論的に安全な一方向変換性」を意味すると同

時に、そのカオス写像を有限精度のコンピュータに実装する場合「カオスの写像」と表現する。従って、「カオスの写像」とは、それが未来に向かう時その情報量は“零ビット”であり、過去に向かう時は“ n ビット”の情報量を必要とすることが「期待出来る」実装技術上のアルゴリズムである。該実装技術上のアルゴリズムは必ずしもカオスの写像であるとは限らないが、カオスの写像アルゴリズムから生み出される鍵を「カオスの鍵；chaotic key」と言う場合がある。

【0023】実施例で示すように、該実装技術では、カオスの写像の未来に向かうその情報量が“零ビット”であることを、暗号／復号を「正確に実行する」根拠とし、認証を「代理鍵」によって「正確に実行する」根拠とし、従って代理鍵を「共有する」根拠とし、また秘密鍵を「一回しか使えない」ように「正確に同期して更新する」根拠とする一方、またカオスの写像が過去に向かうと“ n ビット”の情報を必要とする事を、「情報理論的に安全な秘密鍵」を生み出す根拠にしている。これら全てカオスの鍵の性質に依存する。

【0024】“One-Time Pad Cipher”をネットワークへ適用する際に生じる深刻な鍵管理の問題は、全て情報理論的に安全な秘密鍵の性質を適用して解決した。すなわち、未来に向かう写像アルゴリズムは一意に拡張秘密鍵を決めるという性質を適用して、TCP/IPなどの一般的なプロトコル上に鍵生成システムを実装して「情報理論的に安全な同期制御」を実施しブレイクスルーしたものである。ここで、同期制御とは、

① 鍵ストリーム（keystream）が平文と独立に選ばれることと、

② クライアント・サーバー双方で実行されるカオスの写像をプロトコルが同期して制御する、という二つの意味を代表する。

【0025】この通信プロトコル上の同期制御を「安全な同期プロトコル：Secure Synchronous Protocol：SSP」というが、これによって「認証付きの鍵共有：Authenticated Key Agreement」がオープン・ネットワーク上で可能になる。このユニークな鍵共有メカニズムによれば「鍵を交換するためには、別に安全な通信路を設けなければならない」という“One-Time Pad Cipher”に特有な鍵配送問題を解決できるのである。

【0026】2) 鍵生成システム；有限精度のパソコンで鍵を等確率に作れるか；コントロール集合を生み出す母胎は「連続量」である。ロジスティック曲線のように単純な曲線であるからという理由で、その鍵生成を軽く見る暗号専門家も居るかも知れないが、それはカオス生成の根底にある「連続量とフラクタルな構造」を見落としからである。逆に、このコントロール集合を生み出す構造に注意を払わないで、単にカオス写像をプログラムしたとしても、カオスを作れないことは明らかであろう；連続量とコンピュータの有限精度との差は、宇宙の

大きさとゴミを比較する以上の差がある。従って、有限精度のパソコンで鍵を等確率に作れない。

【0027】しかし、我々は従来のフラクタル構造にプラスして、「加算個の無限を取り扱うフラクタルな演算構造」(注2)をコンピュータの世界に導入することを考案した。

【0028】このフラクタルな演算理論に基づくアルゴリズムは多数考えられるが、どのようなアルゴリズムにしろ「鍵が等確率に出現する」根拠を持った工夫・考案が無ければならない。(特願平8-108058「周期性を考慮したカオスの乱数列の発生装置」及び特願平8-169869「乱数生成装置及び乱数生成システム並びに暗号通信方式」(注2)の乱数生成法は、“Computationally Secureな長期周期軌道”に関する一方、前特願平8-338583「乱数列の発生方法」の乱数生成法は、非周期軌道に関する。

【0029】該アルゴリズムは、いずれもプログラム技術としては多重演算アルゴリズムを利用しているが、それが“Chaotic Orbit”を生成する理論上の根拠に依存している。すなわち「加算個の無限を取り扱うフラクタルな演算構造」を継承してそれに準じている)。

【0030】我々は「鍵が等確率に出現する」ことを期待できる他の工夫も歓迎する。以上要約すると、“Unconditionally Secure”な鍵生成手段とは、秘密鍵とワーク鍵とを生成する手段を備え、かつそれぞれ等確率に生成すること、及びそのプロセスが情報理論的に安全な方向変換性を備えていることである。鍵が等確率に出現する状況証拠には、①“keystream”に自己相関が無い、②0/1ビットの偏りが無い(真正乱数)、演算レジスターに“Lorentz plot”が現れる(実施例参照)などがある。これらは「Perfect Secrecy」を提供する上で考慮されねばならない重要な要因である。

【0031】次の4項でも触れる。

3) 鍵ストリーム；動的ワーク鍵；上記鍵生成システムでは、整数の軌道値を通常10進数で表現する一方、実際暗号に使うワーク鍵には、整数の軌道を圧縮変換した2進のビット列を使う。

【0032】秘密鍵を10進表現で、動的ワーク鍵を2進表現で使うと、この2進ビット列(子)からカオスの軌道(親)を特定しようとしても、言い換えればカオスの写像の内部情報を得たいと盗聴者が意図しても、圧縮変換に伴う 2^{2n} の不確定度と、カオスの写像を過去に遡る 2^i の不確定度が、共に乗算で効いてくるので、

その情報を計算しても得られない、さらに情報理論的に安全になる；その不確定度は「 $2^{2n} * 2^i$ 」である。

【0033】このような情報理論的に安全なワーク鍵を「動的ワーク鍵：Dynamic Work-key」または“secure keystream”と参照する。

【0034】当該鍵生成システムの秘密鍵のEntropyは「 $2n+i$ ビット」である。この不確定度は、丁度 2^{2n+i} 個の“目”を持ったサイコロを振るに等しい。一例として、 $n=64$ 、 $i=128$ と仮定すると、そのサイコロの目は 2^{256} 個ある。該秘密鍵を突き止めるのは、この巨大なサイコロを振るに等しい。

【0035】4) “One-Time Stream Cipher”；対称暗号であるところのストリーム暗号に、上述の情報理論的に安全な鍵生成システムを組み込んで、あらゆる階層の鍵を一回使ったら捨てるとしたルール、つまりプロトコルを導入する。この暗号系を仮に“One-Time Stream Cipher”と言うことにする。

【0036】一般に、ストリーム暗号では、ワーク鍵を平文と同じ長さのビットストリームとして生成すると共に、このワーク鍵と平文とを排他的論理和(XOR)し暗号文にする。

【0037】今“One-Time Stream Cipher”の一つの端末に著目して鍵の解析をして見れば、どの秘密鍵も動的ワーク鍵も、全ての鍵が一度使ったら捨てられる、“One-Time Use”に限られているはずである。この時、当該通信プロトコルに組み込まれている、上記鍵生成システムは、1)でも述べたように秘密鍵を全て等確率で算出し、当該通信プロトコルもそれを等確率で使う(使ったら捨てる)；とするなら、該暗号系は「Perfect Secrecy」を実現していることが期待出来る(注3)。

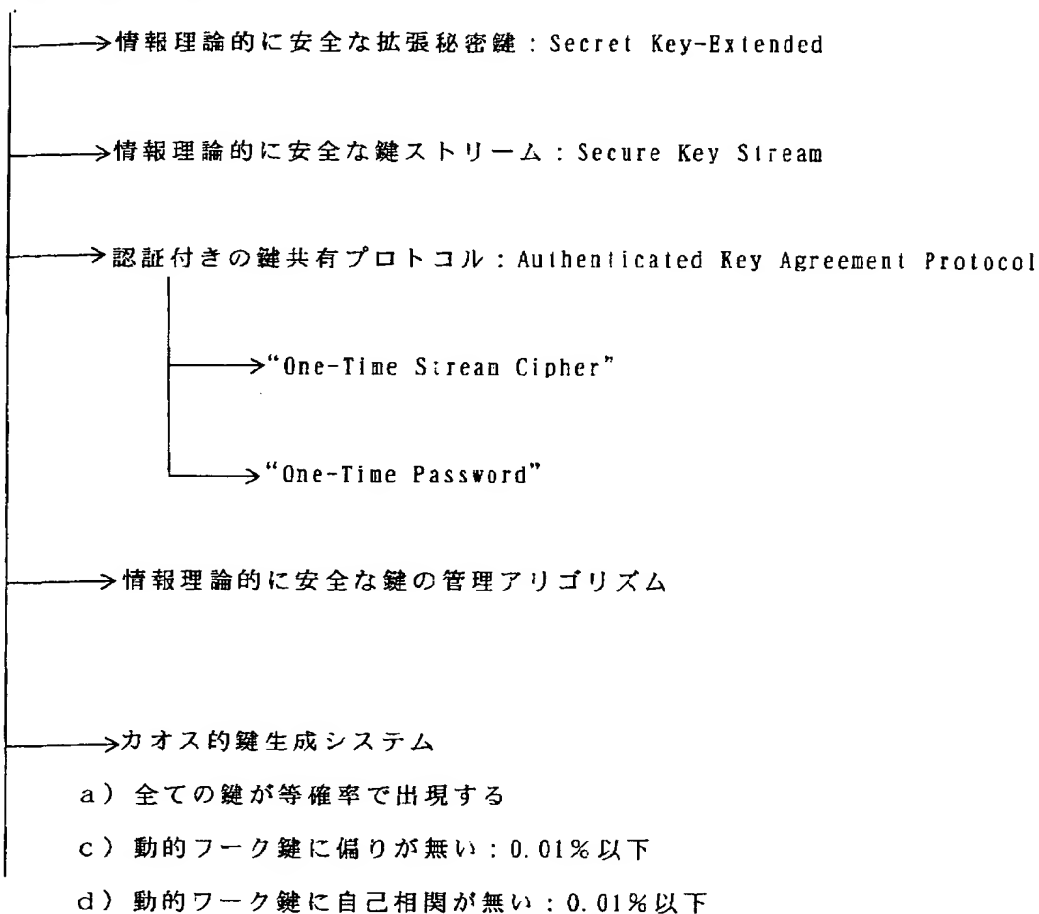
【0038】実際、本発明の“One-Time Stream Cipher”では、生成した秘密鍵を“One-Time use”に限る一方、安定した通信も実現している。この安定した通信は、1)項で紹介したように、端末双方が情報理論的に安全な秘密鍵を用いて「安全な同期プロトコル：Secure Synchro-Protocol」を共有し、秘密鍵を拡張秘密鍵で更新するというダイナミズムを実現しているからである。以上1)～4)項は、「Perfect Secrecy」を実現する暗号系の基礎理論の概要である。

【0039】本特願はこの実装技術に関する：その構成は、

【表1】

【表1】

「perfect secrecy」の実装手段==安全プロトコル；secure
re synchrono-protocol



【0040】その実現手段は以下の通りである；

1. クライアント／サーバー・ネットワーク（以下C／Sネットワーク）などで通信を行う端末双方に、 $2u$ （ $u > 4$ ）進数で演算するレジスターを備え、該レジスターの整数空間 I_s のそれ自身への写像（ $I_s \rightarrow I_s$ ）を実行して「整数の軌道」を生成する写像アルゴリズム部と、該 $2u$ 進数の「整数の軌道」を圧縮変換して2進の「鍵ストリーム」を生成する圧縮変換部、とを備えた鍵生成システムを実装し、

【0041】該鍵生成システムには、その写像アルゴリズムを過去に遡る逆写像を試みると、 n 回の逆写像に対しては“ n ビットのEntropy”が生じだけでなく、上記圧縮変換に伴う“Entropy”もそれに加わる「情報理論的に安全な写像アルゴリズム」が実装されており、該情報理論的に安全な写像アルゴリズムを、「鍵の情報理論的な安全性」を追求する目的で、暗号と復号のアルゴリズムの中に「意図的に」埋め込んだ事の特徴とする情報理論的に安全な対称暗号系。

【0042】2. 記対称暗号系は、該像アルゴリズムによる整数の軌道の任意の軌道値“ ck_0 ”を「秘密鍵」

として共有すると同時に、実際に暗号／復号するワーク鍵には上記「鍵ストリーム」を使う一方、秘密鍵“ ck_0 ”がどちらかの該鍵生成システムにインプットされると、該秘密鍵を情報理論的に安全な秘密鍵にするために、写像アルゴリズムを規定回数（ i 回）実行させて、該秘密鍵“ ck_0 ”のカオスの軌道（ $ck_1, ck_2, ck_3, \dots, ck_i$ ）を作らせると共に、該カオスの軌道値を該端末内に「保存」という規約と、

【0043】さらに、この保存されている i 番目のカオスの軌道値“ ck_i ”すなわち「拡張秘密鍵」を初期値とした新たなカオスの軌道（ $ck_i, ck_{i+1}, \dots, ck_{i+j}$ ）を生成させて、この後者の整数軌道を圧縮変換した「鍵ストリーム」をワーク鍵に使うという規約と、これら二つの規約から構成される「安全な同期プロトコル：Secure Synchronous Protocol：SSP」を実装し、

【0044】該「安全な同期プロトコル」を端末群が共有する事により、対称暗号系の共通鍵の安全性が計算量の安全性“Computational Security”ではなく、情報理論的に安全性“Unconditional Security”の問題に帰着

させたことを特徴とする対称暗号系。

【0045】3. 秘密鍵<c k0>を共有している前記1、2項記載の対称暗号系は、互いに相手を確認しうる手段(ID)として共有している秘密鍵<c k0>を、認証の手段としてネットワーク上で交換せず、保持すると共に、対称暗号系のサーバー上で他の偶然的イベントを初期値とした別の整数の軌道値を生成し、これを前記1項の安全な同期プロトコル(SSP)の下でサーバーが保持しているクライアントの秘密鍵<c k0.serv>によって暗号化し、該秘密鍵<c k0.serv>の「代理の鍵<s k i .serv>」として、他方のクライアントに配送する一方、

【0046】この暗号データを受け取ったクライアントも、該クライアントが保持していた秘密鍵<c k0.clin>により同じく“SSP”の下で該暗号データを復号化して「代理鍵」<s k i .clin>を受け取るという規約「鍵の同期配送プロトコル」を実装し、

【0047】さらに、サーバーは代理鍵<s k n .serv>を初期値としたカオスの軌道値を生成して該拡張代理鍵群を保持し；

【0048】

【数5】 { s k (n+1) .serv, s k (n+2) .serv, s k (n+3) .serv, . . . }、

【0049】一方クライアントも代理鍵<s k n .clin>について、それを初期値とした整数の軌道値を生成して該拡張代理鍵群を保持し；

【0050】

【数6】 { s k (n+1) .clin, s k (n+2) .clin, s k (n+3) .clin, . . . }、

対応する拡張ワーク鍵の一致 : <w k 3 .serv> = <w k 3 .clin>

= <s k n .serv> = <s k n .clin> : 代理鍵を共有し

= <c k i .serv> = <c k i .clin> : 拡張秘密鍵を共有し

= <c k0 .serv> = <c k0 .clin> : 秘密鍵を共有し

= 認証完了

【0059】

などが成立する「鍵の同期認証プロトコル」を備え、

同時に

= <s k n> ; を共有する、

すなわち、

= 代理鍵の情報理論的に安全な配送を完了する

とした「鍵の同期配送プロトコル」を実装し、該「鍵の同期配送プロトコル」と「鍵の同期認証プロトコル」の両者から構成される「認証付きの鍵共有プロトコル: Authenticated Key Agreement Protocol」を実装する事を特徴として秘密鍵の共有を実現する前記1、2項記載の対称暗号系。

拡張ワーク鍵の一致 = 拡張秘密鍵を共有し

= <c k i .serv> = <c k i .clin> ; i = 1-任意

= <c k0 .serv> = <c k0 .clin> ;

= <c k0> ; 秘密鍵を共有する、

【0051】これら拡張代理鍵群をそれぞれ前記鍵生成システムの圧縮変換部に入力し、それぞれ対応する拡張ワーク鍵系列を生成し、かつ保持し；

【0052】

【数7】 {、w k 1 .serv, w k 2 .serv, w k 3 .serv, . . . }

{、w k 1 .clin, w k 2 .clin, w k 3 .clin, . . . }

【0053】これら情報理論的に安全な拡張ワーク鍵群をネットワークへ出して交換する規約「鍵の同期認証プロトコル」を実装し；すなわち、

(1) クライアントからサーバーへ最も新しい「拡張ワーク鍵」<w k 3 .clin>を配送する。

【0054】(2) サーバー上では下記の比較をする。

【0055】クライアントのワーク鍵<w k 3 .clin> : サーバー上のワーク鍵<w k 3 .serv>

(3) 一方サーバーからもクライアントへ比較的に新しい「拡張ワーク鍵」を配送する。この鍵には<w k 1 .serv>又は<w k 2 .serv>どちらを使ってもよい。

(4) クライアント上では(3)項の鍵を使って下記の比較をする。

【0056】サーバーのワーク鍵<w k 1 .serv> : クライアント上のワーク鍵<w k 1 .clin>

(5) 一致を見なかった場合は、該端末の方のイニシャティブで通信を切断する。

【0057】(6) 一致を見た場合、

【0058】

【数8】

【0060】4. 対称暗号系の端末双方が“login”

後、前2項記載の安全な同期プロトコル(SSP)の下で前3項記載の「認証付き鍵共有」を成した後に於いては、当該対称暗号系の中の鍵群は、

【0061】

【数9】

【0062】を無条件に満足する鍵群となるので、“login”が発生する度に、端末双方とも一回使用した秘密鍵を、該鍵群の中の未使用の任意の拡張秘密鍵もって「更新する」ことを特徴とした対称暗号系又は、前記1,2,3項記載の構成要件を満たす“One-Time Stream Cipher”同じく前記1,2,3項記載の構成要件を満たすことを特徴とした“One-Time Password”。

【0063】5. 多数のクライアントの秘密鍵のファイル「鍵箱」を備えたサーバーが、その鍵箱を情報理論的に安全なセキュリティで守るために、前2項記載の安全な同期プロトコル(SSP)の下で前3項の「認証付き鍵共有」を成した後に於いて、該鍵サーバーは、「鍵箱」の任意の秘密鍵 $\langle ck0 \rangle$ を前2項記載の当該秘密鍵 $\langle ck0 \rangle$ の「鍵ストリーム」 $\langle wk - ck0 \rangle$ によって該鍵 $\langle ck0 \rangle$ 自身を暗号化し、

【0064】該「鍵箱」を暗号ファイルCRYPTO($ckn : n=0,1,2,3, \dots$)として管理する一方、前記1乃至4項記載の対称暗号系がこれを復号化するには、該クライアントが保持している鍵 $\langle ck0.clin \rangle$ から生成した上記鍵ストリーム $\langle wk - ck0.clin \rangle$ しか復号化の手段が残されていないので、前記1乃至4項記載の対称暗号系は次の“login”時に、該クライアントの該鍵ストリームをサーバーに配送し、該サーバー上で暗号ファイルCRYPTO($ckn : n=0$)を復号化し、当該クライアントの秘密鍵 $\langle ck0.serv \rangle$ をサーバーへ与えるとした「鍵管理プロトコル」を実装する一方、該対称暗号系は、該サーバーが該秘密鍵 $\langle ck0.serv \rangle$ を入手した後には、前3項記載の規約「認証付き鍵共有プロトコル」の実行に入ることを特徴とした鍵管理アルゴリズム及び鍵サーバー。

【0065】6. 前記1項記載の鍵生成システムのレジスターは、 10^n ($n \geq 1$) 個または 2^n 個 ($n \geq 1$) の桁から成立する一様レジスターであり、その一桁も又 10^n 個または 2^n 個の桁から成立する($10^n \leq 10^n$) 一様レジスターであり、というように「入れ籠

平文Xのビットストリーム長=暗号文Yのビットストリーム長、
=ワーク鍵wkのビットストリーム長、

$$\text{Bit-length}(X) = \text{Bit-length}(Y) = \text{Bit-length}(wk) \quad \text{Eq. 1}$$

$$\text{暗号文} Y = \text{XOR}(\text{平文} X + \text{ワーク鍵} wk) \quad \text{Eq. 2}$$

これを元の平文に戻すには、同じワーク鍵wkを使つて、

$$\text{平文} X = \text{XOR}(\text{暗号文} Y + \text{ワーク鍵} wk) \quad \text{Eq. 3}$$

【0072】ここで注目しなければならない事は、関数関係式Eq. 1 & Eq. 2にワーク鍵wkが入っているのでEq. 3の復号が成立するという自明な関係である。情報理論の視点から見ると、一般的に平文Xと暗号文Yとは互いに「独立な事象ではない」という関係である。これは、そもそも暗号・復号を成立させる自明な、そして必須な関係であることを記憶しておかなければならない。これは対称暗号系の欠点を意味する一方、そもそも暗号

構造」を成した「自己相似形レジスター」であることを特徴とする一方、該鍵生成システムの写像アルゴリズムは、該一様レジスターを 10^n 進数または 2^n 進数の整数空間Isと見なす該整数空間のそれ自身への写像Fである；($F : Is \rightarrow Is$)。

【0066】注1) C.E.Shannon. “Communication theory of secrecy systems” Bell Systems Technical Journal, 28(1949), 656-715.

注2) 特願平8-108058

「周期性を考慮したカオスの乱数列の発生装置」及び特願平8-169869

「乱数生成装置及び乱数生成システム並びに暗号通信方式」

[18] / / / [0024]

特願平8-338583「乱数列の発生方法」

注3) Douglas R. Stinson “CRYPTOGRAPHY: Theory and Practice” Theorem 2.4

注4) 山口昌哉「カオスとフラクタル」ブルーバックス

【0067】

【発明の実施の形態】請求項1, 2, 3, 4, 5, 6記載の対称暗号系は、完全守秘性の実現“Perfect Secrecy Cryptosystem”を意図した。該暗号系を「使い捨てのストリーム暗号システム: One-Time Stream Cipher」としてインターネットへテスト実装した。その実施の形態を述べる。

【0068】1. 完全守秘性をオープン・ネットワークへ実装する立場から、まず一般的なストリーム暗号システムについて再検討する。

【0069】図-1にて、ストリーム暗号システムは、平文Xをビットストリームであるワーク鍵によって暗号文Yに変換する：関数関係式で表すと、

【0070】

【数10】

【0071】

【数11】

系の安全性を計算量の安全性に持ち込まなければならなかった理由でもある。

【0073】一方、情報理論的な安全性を追求するところの完全守秘性とは「暗号文Yが与えられた時に平文Xを推定する確率と平文Xそのものが生じるアприオリ確率とは等しい」時の安全性を述べた理論である。この安全性とは、暗号文Yが与えられた時に平文Xを推定するのは丁度、サイコロの目が“星の数ほど有る”ようなサ

イコロを振って、そしてそのサイコロは誰でも手に入れようとするれば入るものであるが、次にどの“目”が出るか誰も予想する者が居ないということを述べているのと同じである。

【0074】この完全守秘性は、次のEq. 4とEq. 5が

$$\begin{aligned} \text{Prob. } P(\text{平文 } X \mid \text{暗号文 } Y) &= \text{Prob. } P(\text{平文 } X) & \text{Eq. 4} \\ \text{Prob. } C(\text{暗号文 } Y \mid \text{平文 } X) &= \text{Prob. } C(\text{暗号文 } Y) & \text{Eq. 5} \end{aligned}$$

【0076】従って、上記条件「平文Xと暗号文Yとが独立変数である」とは、一般的な関数関係式Eq. 1 | 2 | 3そのものを否定するような主張である。

【0077】しかし、この時、ワーク鍵が平文Xと独立に選ばれ、かつワーク鍵の情報理論的な性質が「どのような平文X、どのような暗号文Yに対してもEq. 2を成立させる鍵が唯一存在し、同時に全ての鍵が等確率に使われるならば、該平文Xと暗号文Yとは完全守秘性の主張、Eq. 4&Eq. 5を満たす」ということが証明されている：詳細な条件に関しては、注1&注3に譲る。

【0078】この定理が、どのようにして一般的なストリーム暗号システム (Eq. 1からEq. 3) に実装されたか、以下順に述べるところである。

【0079】II. まず最初に、当該ストリーム暗号システムの構成要件を図-1に示す；図中に於て、クライアント/サーバー・ネットワーク（以下C/Sネットワーク）などで通信を行う端末双方に、 2^u ($u > 4$) 進数で演算するレジスタを備え、該レジスタの整数空間 I_s のそれ自身への写像 ($I_s \rightarrow I_s$) を実行して整数の軌道を生成する写像アルゴリズムと、 2^u 進数の該カオスの軌道を圧縮変換して2進ビット列の「鍵ストリーム」を生成する圧縮変換部とを備えた鍵生成システムを実装する一方、該端末双方は、該整数の軌道系列の任意の軌道値 $\langle c k 0 \rangle$ を「秘密鍵」として共有すると同時に、該秘密鍵を初期値として生成した整数の軌道をさらに圧縮変換したところの上記2進ビット列を平文/暗号文と等量作り<ワーク鍵>として使う。図中、秘密鍵 $\langle c k 0 \rangle$ が共有されているので、該秘密鍵から作られたワーク鍵 w_k も正確に等しい (The Entropy = “0” Bit に示される)。これ等がストリーム暗号システムを構成する (前々特願請求項2、3、に相当する) (注2)。該ス

$$\begin{aligned} \text{拡張秘密鍵} &= F_i (\text{秘密鍵}) \\ \langle c k i \rangle &= F_i (\langle c k 0 \rangle) \end{aligned}$$

【0083】 F_i : 該写像アルゴリズムは、レジスタが作る整数空間 (I_s) 内の全ての鍵 $c k 1$ 、 $c k 2$ 、 $c k 3$ 、 \dots 、 $c k i$ 、 \dots 、 $c k n$ を等確率で生成するだけでなく、該動的ワーク鍵 (ビットストリーム) に偏りが無い (0.01%以下) 及び該動的ワーク鍵に自己相関がない (0.01%以下) という属性を提供できる非線形関数

$$\text{Dynamic } w_k (C k i + j) = \text{One-way Compr. } F_{i+j} (\langle c k 0 \rangle)$$

One-way Compr : 2^u 進数から2進への一方向圧縮変換

【0086】該動的ワーク鍵 $w_k (C k i + j)$ は、ネッ

同値であることを主張し、その確率等式は「平文Xと暗号文Yとが独立変数である」時に成立する：

【0075】

【数12】

トリーム暗号システムを構成する枠組みの中で、秘密鍵を情報理論的に安全な拡張秘密鍵 “Secret Key—Enhanced” にするプロトコルを図-2に示す。

【0080】該ストリーム暗号システムは、共有している秘密鍵 $\langle c k 0 \rangle$ を直接使わないで上記鍵生成システムにインプットし、該秘密鍵 $\langle c k 0 \rangle$ を初期値として写像アルゴリズムを規定回数 (i 回) 実行させて、該秘密鍵 $\langle c k 0 \rangle$ のカオスの軌道 ($c k 1$ 、 $c k 2$ 、 $c k 3$ 、 \dots 、 $c k i$) を作らせる。この軌道を該暗号システム内に「保存」した後、この i 番目のカオスの軌道値 $\langle c k i \rangle$ を初期値とした写像アルゴリズムを、さらに (j 回) 実行させ、新たなカオスの軌道 ($c k i$ 、 $c k i+1$ 、 \dots 、 $c k i+j$) を生成し、当該平文又は暗号文と等量のワーク鍵 w_k を生成する (j の値は平文/暗号文と等量になるように決める)。図-2中、秘密鍵 $\langle c k i \rangle$ から見て、ワーク鍵 w_k に至るエントロピーは正確に “0” ビットであり、一方、秘密鍵 $\langle c k 0 \rangle$ に至るエントロピーは正確に “ i ” ビットである。故に、秘密鍵 $\langle c k 0 \rangle$ から見て、該秘密鍵 $\langle c k i \rangle$ を「拡張秘密鍵」と呼ぶ一方、該ワーク鍵 w_k から見ると該秘密鍵 $\langle c k 0 \rangle$ に至るエントロピーはさらに増加し、正確に “ $i+j$ ” ビットとなる。これで、より安全なワーク鍵となるので、該ワーク鍵 w_k を「動的ワーク鍵 : Dynamic $w_k (C k i + j)$ 」または “secure keystream” として参照する。これら鍵を総称して、カオスの鍵 “chaotic keys” として参照する場合がある一方、その生成システムを図-3に示す。

【0081】該図-3/S7は、次のEq. 6に対応する；

【0082】

【数13】

$$\text{Eq. 6}$$

でなければならない。“ i ” は重要なコントロール・パラメータである。

【0084】該図-3/S2/S2/S5/S6は、次のEq. 7に対応する；

【0085】

【数14】

$$\text{Eq. 7}$$

トワーク上を移動したり、公開されるような使い方をされる場合も有る、従って盗聴される場合もあるが、該秘

秘密鍵 $\langle c k 0 \rangle$ の安全性は「情報理論的に安全である」。 $i = 128$ として実装すると、該エントロピーは “ $128 + j$ ” ビットである。この不確定度は、サイコロの目が “ 2^{128} 以上” も有るサイコロを振るのと同じである。

【0087】該図-1/図-2/図-3に示される構成要件の中で、対称暗号系の秘密鍵を拘束するEq. 6 Eq. 7に示された規約を「安全な同期プロトコル: Secure Synchro-Protocol」として参照する。該安全な同期プロトコルを実装した対称暗号系では、その共通鍵の安全性が従来の計算量の安全性の問題 “Computational Security” にならず、情報理論的に安全な又は確率論的に安全な問題 “Unconditional Security” に帰着する。その理由を要約すれば、

【0088】① 該安全な同期プロトコル “Secure Synchro-Protocol: SSP” という実装技術が、端末群に共有され、

② 端末双方のカオスの鍵 “chaotic keys” を制御し、

③ その秘密鍵 $\langle c k 0 \rangle$ を情報理論的に安全な拡張秘密鍵 $\langle c k i \rangle$ に変換し、これから動的ワーク鍵 wk ($C k i + j$) “secure keystream” を生成する、からである。

【0089】III. この安全な同期プロトコル (SSP) をパラダイムにして、秘密鍵をどのようにして “One-Time” にするかという本題に入ることになるが、その前提として、鍵をどのようにしてネットワーク上に配送するか、そのためにはクライアント/サーバー双方の認証をどのように実施するか、を示さねばならない。「代理鍵の安全同期配送プロトコルおよび代理鍵による安全同期認証プロトコル」図-4と「初期認証シーケンス/Passnet更新シーケンス」図-6はその仕組みを示す。SSPのパラダイムでは、認証と鍵配送とが同時進行し、内容は同じであるが違った目的に到達するという特徴がある。

【0090】その構成要件は図-2に示すように、秘密鍵 $\langle c k 0 \rangle$ を共有している請求項1、2記載のストリーム暗号システムは、C/Sとも共有している秘密鍵 $\langle c k 0 \rangle$ を、互いに相手を確認しうる手段であるID=秘密鍵をネットワーク上で交換する事をせず、両者とも「保持する」。

【0091】その代わり、図-4、図-5に示すように、該ストリーム暗号システムはサーバー上で他の偶然的イベントをトリガーとした初期値から別のカオスの軌道値 $\langle s k n . serv \rangle$ を生成し、これをサーバーが保持している秘密鍵 $\langle c k 0 . serv \rangle$ の「代理鍵」として、前記2項の安全な同期プロトコルと該秘密鍵 $\langle c k 0 . serv$

$$\langle s k n . serv \rangle = \langle s k n . clin \rangle = \langle s k n \rangle ; \quad \text{Eq. 8}$$

【0099】結局、鍵を共有するようになるので、該ストリーム暗号システムは、鍵の情報理論的に安全な配送を「認証と同時に」完了する「鍵の同期配送プロトコ

ル」によって暗号化し、クライアントに配送する。この偶然的イベントの発生は、いままで知られているどのような手段でもよい；「代理鍵」は一度使用されるとその都度、瞬時に捨てられるから。

【0092】配送の様子を図-6にて、1-秘密鍵DBから7-暗号化された “ $s k n$ ” までに示す。1-秘密鍵DBから該当するユーザーの秘密鍵 $\langle c k 0 . serv \rangle$ をどのように探し当てるか、この仕組みを「初期認証」というが、これには “Passnet” と名付けられたパスワードを使う。 “Passnet” は、ユーザーが携帯するメディア (図-6-9に示す) に、ユーザーの秘密鍵 $\langle c k 0 . serv \rangle$ と一緒に格納されているランダム・データである。

【0093】該 “Passnet” とは、どのようなパスワードか後述する (なおユーザーの携帯するメディアにどのように「安全に格納する」かの仕組み等は派生技術なので除外する) この暗号データを受け取ったクライアントは (図6では初期認証応答として示される)、該端末が保持していた秘密鍵 $\langle c k 0 . clin \rangle$ と請求項1記載の「安全な同期プロトコル (SSP)」により同じく該暗号データを復号化して代理鍵 $\langle s k n . clin \rangle$ を受け取る。

【0094】但し、この段階では「サーバーが保持するクライアントの秘密鍵」と「クライアントが保持している秘密鍵」とは等しいとは限らないので、前者を $\langle c k 0 . serv \rangle$ 後者を $\langle c k 0 . clin \rangle$ と記号して区別している (図-4では $\langle S k ? \rangle$ と表現している)。

【0095】本来、秘密鍵は共有されている一対のID兼用暗号鍵であるが、ネットワーク上でパスワードで引き当てただけでは、それが後述する “One-Time Password” であっても、両者の秘密鍵が同じであるとは保証できない。

【0096】従って、この時点では秘密鍵、代理鍵ともクライアントに所属していたものかサーバーに所属していたものなのか、両者の区別を記号化した；図-6では、 $\langle s k n . serv \rangle$ を7-暗号化される代理鍵、一方 $\langle s k n . clin \rangle$ を12-復号化された代理鍵として区別している。

【0097】このような構成要件の中では、秘密鍵が同じであれば代理鍵も等しいし、その逆に、代理鍵が等しければ秘密鍵もまた等しいことが判る。故に認証のためには秘密鍵の代わりに代理鍵を調べる。もし該暗号化される代理鍵と復号化された代理鍵とが等しいことが、安全に確認出来たとすると、

【0098】

【数15】

ル」を備えた事になる。

【0100】IV. 該Eq. 8を確認する手段を「鍵の同期認証プロトコル」と名付け、以下の図-7と図-4に

説明する。代理鍵を直接ネットワーク上で交換するのは
“insecureとなるから、代わりに、図一7に於いてサー
バー上の代理鍵15<skn.serv>を初期値としたカ
オスの軌道16—

【0101】

【数16】 { sk (n+1) .serv, sk (n+2) .serv, sk (n+3) .serv, 、 、 }

【0102】を生成し、一方クライアント上の代理鍵17<skn.clin>についても、それを初期値としたカオスの軌道系列18—

【0103】

【数17】 { sk (n+1) .clin, sk (n+2) .clin, sk (n+3) .clin, 、 、 }

【0104】を生成し、これらカオスの軌道系列を前記乱数生成システムに戻して該圧縮変換部に入力し、それぞれ対応するワーク鍵系列に変換する；

【0105】

【数18】 (、wk1.clin, wk2.clin, wk3.clin, 、 、)； 図一7—22/23/24

【0106】及び

【数19】 (、wk1.serv, wk2.serv, wk3.serv, 、 、)； 図一7—19/20/21

【0107】これらワーク鍵系列は「情報理論的に安全な拡張ワーク鍵群」である。従って代理鍵を直接ネットワーク上で交換する代わりに、その拡張ワーク鍵をネットワークへ出して交換する事が出来るようになる。該拡張ワーク鍵を初期認証に使う場合、図一6では“Passnet”と呼び、通常の“Password”と区別した。交換のプ

拡張ワーク鍵の一致：<wk3.serv> = <wk3.clin>

= <skn.serv> = <skn.clin>：代理鍵の共有し

= <cki.serv> = <cki.clin>；拡張秘密鍵の共有し

= <ck0.serv> = <ck0.clin>；秘密鍵の共有し

= 認証完了（図一4に示す） Eq. 9

【0114】そして、同時に前述のEq. 8を成立させる；

= 代理鍵の情報理論的に安全な配送

= skn：代理鍵を共有する； Eq. 8

【0116】故に「鍵の同期配送プロトコル」と「鍵の同期認証プロトコル」とは同じ手続きに基づく二つの異なる適用であることが判る。これはカオスの写像の安全な同期制御に基づく、あるいは安全な同期プロトコル

(SSP)に基づき鍵共有の新しい方式を与える。「鍵の同期配送プロトコル」と「鍵の同期認証プロトコル」の両者を合わせて、このスキームを「認証付き鍵共有プロトコル：Authenticated Key Agreement Protocol (AKAP)」として参照する；すなわち、

【0117】

【数22】AKAP = SSTP + SSAP

SSTP：Secure Synchro-Transfer Protocol

プロトコルは比較的に新しい「拡張ワーク鍵」<wk3.clin>を先に使う（この理由も既に述べた）。

【0108】すなわち、

(1) クライアントからサーバーへ最も新しい「拡張ワーク鍵」<wk3.clin>を配送する—25。

【0109】(2) サーバー上では26—下記の比較をして一致していれば「クライアントの認証」を行う：クライアントのワーク鍵<wk3.clin>：サーバーのワーク鍵<wk3.serv>サーバーがクライアントの認証を実施した場合、その確認のサインをどのような手段でクライアントへ通信するか、通常のTCP/IPのプロトコルを使うのは“insecure”となるのは明らかである。そこで、

(3) サーバからクライアントへも「拡張ワーク鍵」を配送する。この鍵には拡張ワーク鍵であれば<wk1.serv>又は<wk2.serv>どちらを使ってもよい。

【0110】図一5/2では、<wk1.serv>を使っている—21。

【0111】(4) クライアント上では—26の比較をして次のことを確認する：

①クライアントはサーバ上でクライアントの認証が実施された事を確認する；

②クライアントがサーバを認証する；

(5) 一致を見なかった場合は、該端末の方のイニシャティブで通信を切断する—27。

【0112】この鍵の同期認証プロトコルでは、

【0113】

【数20】

【0115】

【数21】

SSAP：Secure Synchro-Authentication Protocol

【0118】該認証付き鍵共有プロトコルは種々の目的に応用出来る。これをサーバーの視点で考えると、従来に無い優れた特徴を備えた「鍵サーバー：key server」の機能を与える。

【0119】V. “One-Time Password”；代理鍵とそのワーク鍵に関する安全な同期制御(SSP)よれば、

【0120】

【数23】Eq. 8/Eq. 9 = その他の拡張ワーク鍵群を共有する、Eq. 10

【0121】も同時に実現する。該「その他の拡張ワーク鍵」—Eq. 11も共通鍵であることに着目すると、

次回クライアントがサーバーへ“loginする際のパスワードとして使える事を発見する。しかも該拡張ワーク鍵は、代理鍵の代わりをする、情報理論的に安全な「公開鍵」であるだけでなく、クライアントがサーバーへ“loginする度に、該サーバー上で他の偶然的イベントから生成した整数の軌道値であったことを思い出すと、該拡張ワーク鍵が“One-Time”であることは明らかである。

該拡張ワーク鍵を“One-Time Password”にする手順

$\langle wk0 \rangle \leftarrow \langle wk2 \rangle$

【0124】該ワーク鍵 $\langle wk2 \rangle$ が次回使われる“Passnet”である。

【0125】こんな簡単なプロトコルで“One-Time Password”が成立するのは、前記の認証付き鍵共有プロトコルのお陰である：図-8/検証-1に示されるシステムはインターネット上に公開されているので、実際に鍵サーバーへ“loginを実施して“One-Time”検証出来る；“loginをインターネット上で連続して実施し“One-Time”の“Passnetを検証する；（http://www.meteora.amadagp.co.jp/）。

【0126】VI. “One-Time”のセッション鍵系列；ここで共有された代理鍵は、種々のアプリケーションを可能にする；Eq. 8より

【0127】

【数25】 $\langle skn.serv \rangle = \langle skn.clin \rangle = skn : n = 0 \sim i$

【0128】これは拡張共通鍵群であるから、

① 該代理鍵はクライアントとサーバーが暗号セッションを持ったという証拠に使える。これは、商取引きの事実を証明する、保存に値するデータである：“Non-repudiality in Electric Commerce。”

【0129】② 該クライアント・サーバー双方は、安全な同期プロトコル（SSP）を介して“One-Time”のセッション鍵を「等確率に」、「連続して」、「ほとんど無数に」共有する。該拡張共通鍵群を多数共有している様子を図-5に示す。該“One-Time”のセッション鍵

拡張秘密鍵の一致 $= \langle cki.serv \rangle = \langle cki.clin \rangle ; i = 1 \sim \text{任意}$
 $= \langle ck0.serv \rangle = \langle ck0.clin \rangle ;$
 $= \langle ck0 \rangle ; \text{秘密鍵を共有する；}$

Eq. 9

【0134】を無条件に満足する。この事は、「TCP/IPなどの一般的なプロトコルを介しても」、認証付き鍵共有を経過することにより、「写像アルゴリズムの安全な同期制御」をリモートで正確に実施できると同時に、システムを“fail-safe”に設計出来ることを意味する。

【0135】よって、“login”する度に、拡張秘密鍵をもって秘密鍵を正確・確実に更新出来る手段を得ること

$\langle ck0 \rangle \leftarrow \langle ck1 \rangle$

【0137】該更新のタイミングは、前述図-7-28/29の“Passnetの更新処理と同じ”である。この様な単純なプロトコルの適用で対称暗号系を“One-Time”に

を、wk2 “Passnetの更新処理；図-7-28/29に示す。

【0122】図-6-9のディスクに示される“Passnetとは、初めの“loginに使われた $\langle wk0 \rangle$ である；これを図-7-28/29で、新しい拡張ワーク鍵 $\langle wk2 \rangle$ と入れ替える；

【0123】

【数24】

Eq. 11

の使い道には、例えば、特定のクライアントだけに特定のコンテンツやバリュエータを提供するためのIDの機能を持たせるアプリケーションが有る。（なお、図-9/検証-2のシステムはインターネット上で検証出来る；（http://www.meteora.amadagp.co.jp/を参照ください）。三番目に、

③ 該サーバー（鍵サーバー）に参加しているクライアント（A）と他のクライアント（B）に共通のセッション鍵を安全に与えることが出来る。一度設定されると、該鍵サーバを介さないでも、クライアント（A）—（B）は“One-Time”のセッション鍵系列を「ほとんど無数に」共有する。このアプリケーションは従来の“Kerberos”などに見られない便宜をユーザーに与える。

【0130】④ 鍵サーバー、セッション鍵、該セッション鍵から作った“Passnetなどが“One-Time”のデジタル署名技術を構成する。

【0131】これら具体的な適用事例の紹介は、認証付き鍵共有（AKAP）の派生技術であるので、省略し請求から外した。

【0132】VII. “One-Time Stream Cipher”；安全な同期プロトコル（SSP）をベースにした「認証付き鍵共有」を成した後に於いては、当該ストリーム暗号システムの中の拡張秘密鍵は

【0133】

【数26】

とになった。該拡張秘密鍵の一つ $\langle ck1 \rangle$ （図-2に示される）は、そのエントロピーが“i-1”ビットであるので、元の秘密鍵 $\langle ck0 \rangle$ の次に情報理論的に安全な鍵である。よって秘密鍵の更新処理は極めて単純となる；

【0136】

【数27】

Eq. 12

出来るのは、該暗号系が認証付きの鍵共有プロトコルによって成立しているからである。

【0138】VIII. 鍵管理と鍵サーバー；オープンネッ

トワークでは、一般に第三者の信頼できる機関 (truste d authority:TA) または証明機関 (certificate author ity:CA) を必要としている。ユーザの身分を証明したり、ユーザの鍵を送付したりする。鍵を管理するという事は、これだけの適用に止まらない。鍵を生成し管理するというこのTA、CAこそ“データに価値を与える”要である。

【0139】つまり署名技術の核心とその適用を見抜くならば、TA、CAを第三者の“信頼できる人”に期待するのではなく、“第三者の信頼できるシステム”にしなければならぬことが判る。丁度、紙幣や小切手を人手に頼ってガリ版で刷っているような状況を想像してみれば良い。

【0140】請求項5の発明は、鍵の運用管理に関してクライアント／サーバーという当事者以外に一切人を介在させないこと、及び鍵を情報理論的に安全なセキュリティで守る仕組みに関する。サーバーは多数のクライアントの秘密鍵のファイル「鍵箱」を備える；図-6、秘密鍵DBがそれに該当する。秘密鍵DBの管理を次のように行う；サーバーは、前1、2項記載の安全な同期プロトコル (SSP) の下で前3項の「認証付き鍵共有」を成した後に於いて、鍵箱の任意の秘密鍵<ck0>を前1、2項記載の当該秘密鍵<ck0>の「鍵ストリーム」<wk-ck0>によって該鍵<ck0>自身を暗号化する。従って、該「鍵箱」は暗号ファイルCRYPTO (ckn:n=0,1,2,3,...) として管理される一方、該サーバーは「復号化の手段を失う」。

【0141】該復号化の手段を持っているのはクライアントだけである。すなわち、クライアントが保持している鍵<ck0.clin>から生成したところの前1、2項記載の「鍵ストリーム」、<wk-ck0.clin>しか復号化の手段が該システムに残されていない。そこで、前記1乃至4項の対称暗号系は次の“login”時に、クライアントの該鍵ストリーム<wk-ckclin>をサーバーに配送し、該サーバー上で暗号ファイル CRYPTO (ckn:n=0) を復号化し、当該クライアントの秘密鍵<ck0.serv>をサーバーへ与えとした「鍵管理アルゴリズム」を実装する：

(1) 初期認証時 (図-6-1、9→1) に於いて、“passnetの送付と共に、上記鍵ストリーム<wk-ck0.clin>を送付する。

(2) 該サーバーは、該秘密鍵<ck0.serv>を入手する。

【0142】ここで、暗号化以前の<ck0>と復号化後の秘密鍵<ck0.serv>とは必ずしも等しくないことに注意する。

【0143】ここからは、先のIII—代理鍵の安全同期配送プロトコル、及びIV—代理鍵による安全同期認証プロトコルを実施する。言い換えると、前3項記載の規約「認証付き鍵共有」を成して前記1乃至4項記載の対称

暗号系を成立させるのである。

【0144】このように、極めて単純なアルゴリズムで、クライアント／サーバーという当事者以外に一切人を介在させない鍵の運用管理及び鍵サーバーを実現出来るようになる。

【0145】IX.フラクタルな演算構造

請求項1及び6記載の鍵生成システムは、コントロール集合を生む従来のフラクタル構造以外に、有限精度のコンピュータの中に導入された新たなフラクタルな演算構造を持つ。その結果を図-10、図-11に示す。この事例では、非線型関数に、ロジスティックス曲線を使っている。該自己相似形レジスターの仕様は：

一様レジスターの桁数；1000個：10³、

入れ籠レジスターの桁数；1000個：10³、

一様レジスター上の演算は、10³を“法”とする。

【0146】該自己相似形レジスターにより、連続量である線分[0、1]が“discreteな1000個の微少区間に分けられ、その微少区間がさらに1000個の微少区間に分けらるという風に順次考えることが可能となる。その結果連続量の属性の一つである[加算個の無限]を“discreteな自己相似形アルゴリズムで扱えるようになる。この時のフラクタル次元は従来の“0.63”から“0.63+1.0”に上がる (該理論は他に譲る、ここではその様子を実施例、図-10、図-11で示した)。図-10のプロットは、1500回程度の写像演算をした結果の、一様レジスターの先頭桁の数値を表す；これが、ロジスティックス曲線を意味することは明らかである。同時に、図-11は、該先頭桁の次の桁の数値を表すプロットである。これ以降の桁の数値も、全て該図-11のプロットのような様相を示す (省略する)。この乱雑なプロット図-11と先の整然としたプロット図-10の意味するところは、乱雑な方のプロットが“Lorentz plot”注⁴に相当する。一見乱雑に見える現象 (図-11) と (例えば、お湯の沸騰状況がこのような“Lorentz plot”として現れるが)、その現象の背後には整然とした法則性が存在しているという関係、この場合ロジスティックス曲線 (図-10) が存在しているという関係を、プロット図-11と図-10が示したのである；一見乱雑に見える現象 (図-11) とその背後にある法則 (図-10) とのシミュレーションである。

【0147】請求項6記載のフラクタルな演算構造とカオスの写像アルゴリズムが、カオスの軌道を成すという状況証拠としては、図-10&図-11以外に、鍵ストリームの自己相関が0.01%以下、1/0ビットの偏りも0.01%以下というデータに現れている。

【0148】

【発明の効果】発明の実施の形態に於いて、“One-Time Stream Cipher”をオープン・ネットワークの完全守秘性暗号系“Perfect Secrecy Cryptosystem”として紹介

した。

【0149】従来、“one-time pad cipher”は完全守秘性暗号系であるにもかかわらず、オープン・ネットワークへ適用出来なかった。“One-Time Stream Cipher”はその困難を解決した。その発明された問題解決法とは、一つは情報理論的に安全な鍵生成システム“Secure Key Stream”と情報理論的に安全な同期プロトコル“Secure Synchronizing Protocol”である。その上に築かれている、もう一つの発明が、認証付き鍵共有プロトコル“authenticated key agreement protocol”である。実は“One-Time Pad Cipher”に必要であった「もう一つ別の安全な通信路」とは、この鍵共有プロトコルがその仕組みを提供しているのである。例えば、インターネット上で“One-Time Pad cipher”を実現するという事は“One-Time Pad cipher”と同等な安全性を帯びている「認証技術」も同時に備えねばならないことを意味している。しかし、ここにRSAは使えない；この安全性は計算量の安全性“Computational Security”であるから。

【0150】従って、当該発明はインターネット上で使える情報理論的に安全な認証技術を発明したという主張に等しい；それをストリーム暗号系に実装し、“One-Time Stream cipher”と名づけたのである。

【0151】この視点から見ると、認証付き鍵共有プロトコルを実装したサーバーは「情報理論的に安全な鍵サーバー」である。もし、従来のプロキシサーバーに該鍵サーバーを実装したらどうなるか；該鍵サーバーは使用回数が無制限の“One-Time Password”をサービスする。同じく、wwwサーバーに実装すれば、該wwwサーバーは、①認証機能“One-Time Password”を備え、②イントラネットのDBを暗号DBに成し、その暗号と復号の機能を提供し、③インターネットとイントラネッ

トをシームレスに統合する新しいオープンネットワークを提供する。

【0152】さらに、該鍵サーバーの下に金融証券のアプリケーションを置いたらどうなるか。当然、該金融機関が該鍵サーバーを管理するから、該システムは、従来の金融機関の基本システムを維持したまま、しかもオープン・ネットワーク上でのトレードや決済を可能にする。将来、金融システムにも適用可能な暗号系とは、このような鍵サーバーを含んだ完全守秘性暗号系になるはずである。

【図面の簡単な説明】

【図1】ストリーム暗号の生成を説明する説明図である。

【図2】拡張秘密鍵と情報理論的に安全なストリーム暗号システムを説明する説明図である。

【図3】カオス的鍵と鍵ストリームの生成・更新を説明するフローチャートである。

【図4】代理鍵の安全同期配送プロトコルと代理鍵による安全同期認証プロトコルを説明する説明図である。

【図5】One-timeのセッション鍵群を説明する説明図である。

【図6】初期認証シーケンスを説明する説明図である。

【図7】更新シーケンスを説明する説明図である。

【図8】検証1を説明する説明図である。

【図9】検証2を説明する説明図である。

【図10】ロジステック曲線の説明図である。

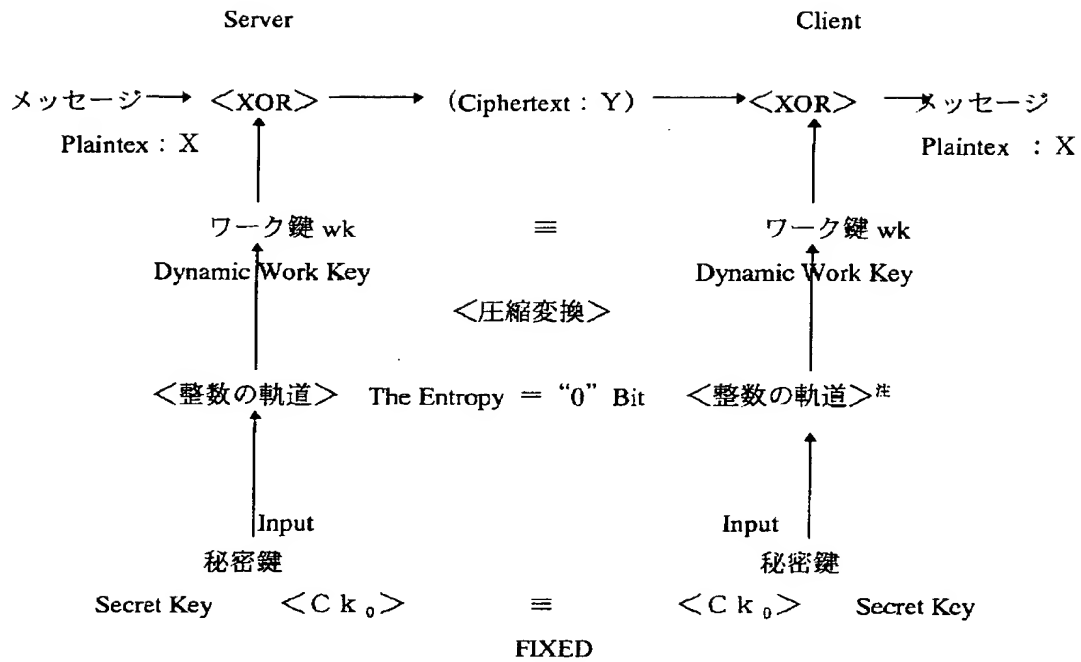
【図11】本発明による演算結果を説明する説明図である。

【符号の説明】

Sk o セッション鍵
Wk ワーク鍵

【図 1】

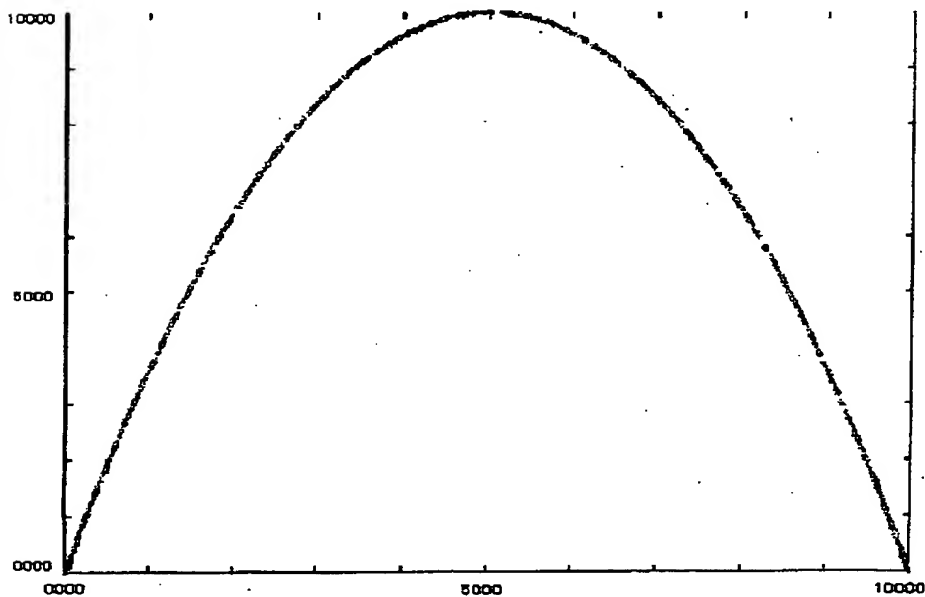
ストリーム暗号

Stream Cipher

注)

$\langle \text{整数の軌道} \rangle$: 秘密鍵 $\langle Ck_0 \rangle$ を初期値とした軌道を指す :

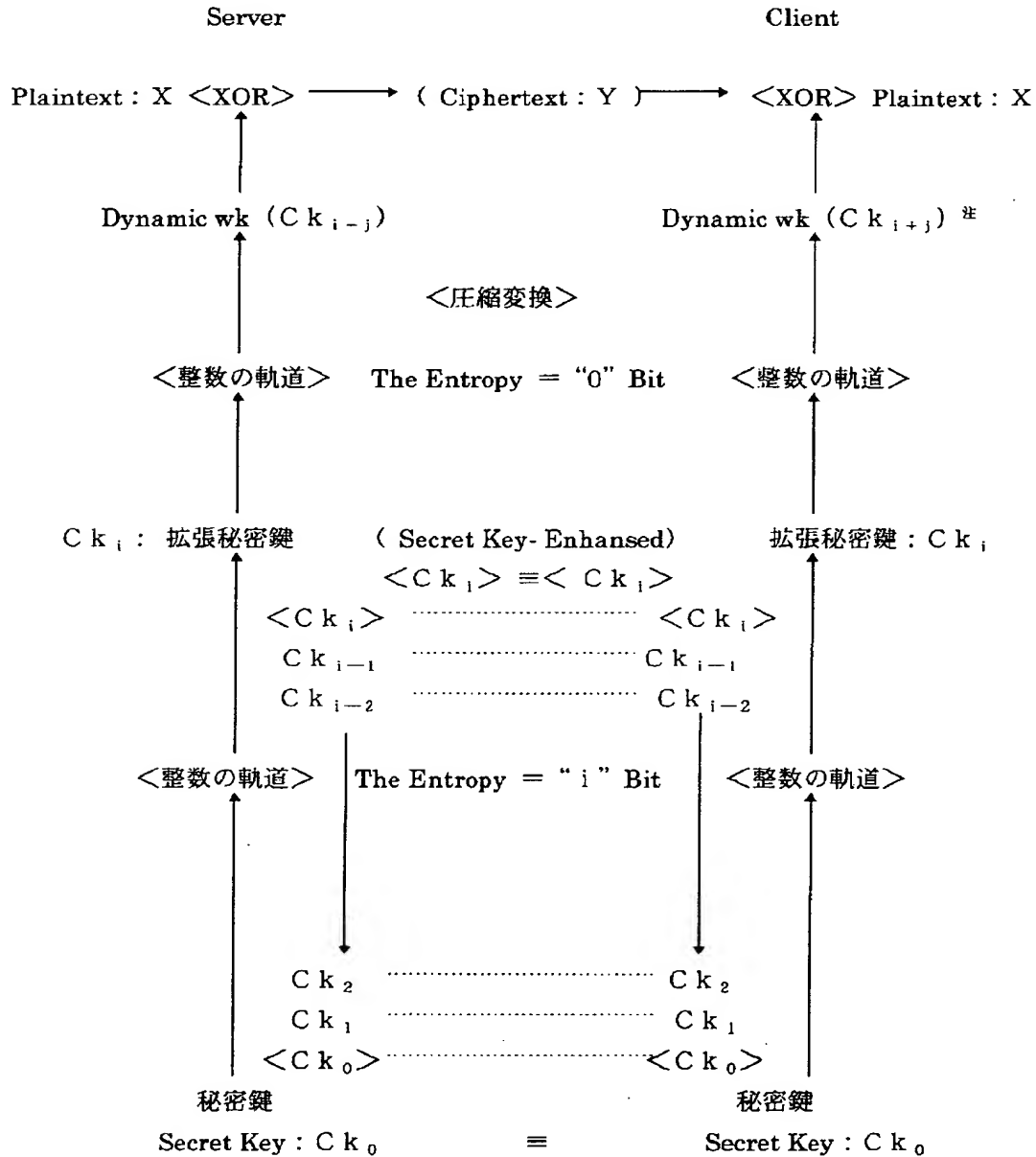
【図 10】



【図2】

拡張秘密鍵と情報理論的に安全なストリーム暗号システム

< Secret Key_Enhanced and Unconditionally_Secure Stream Cipher System >

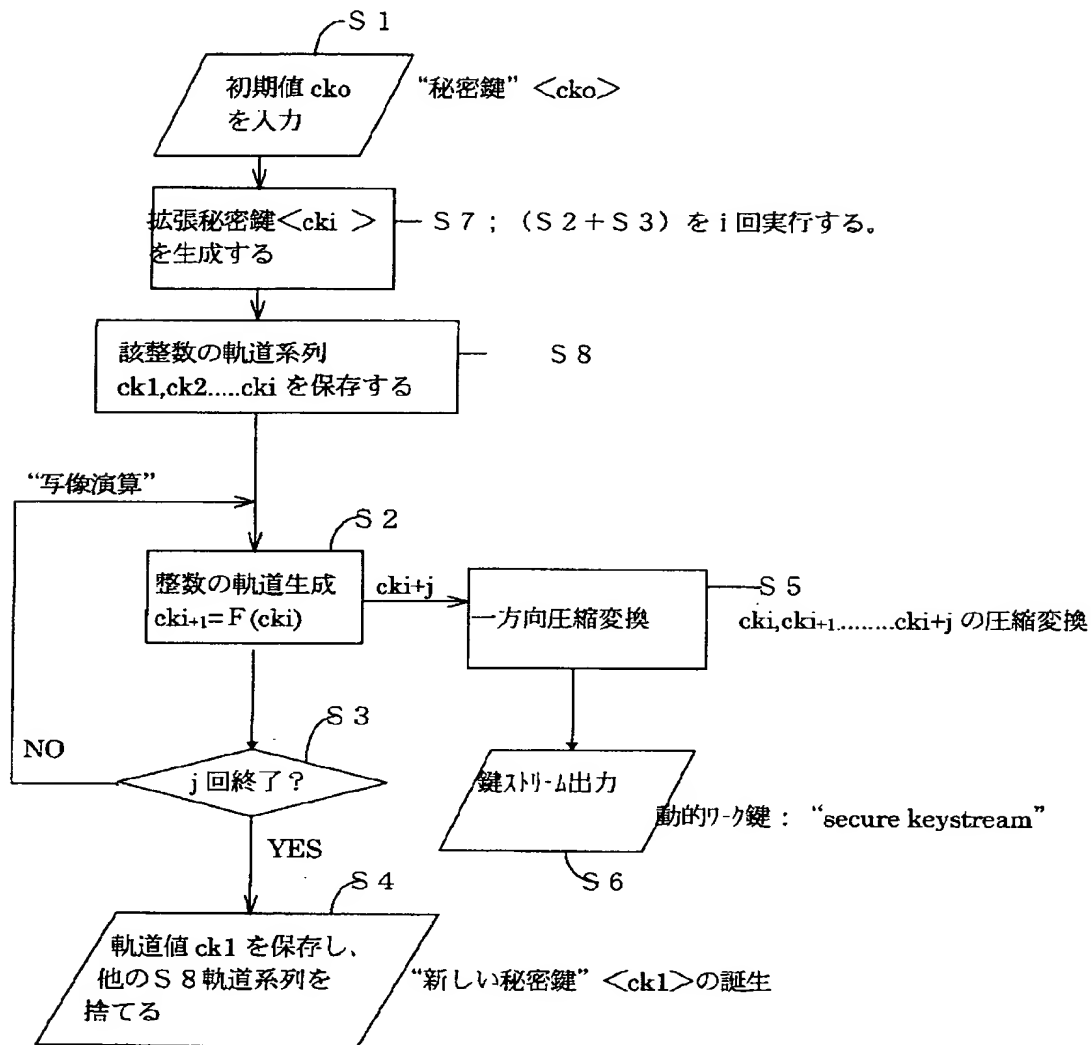


注)

Dynamic Wk (Ck_{i+j}) : $\langle Ck_i \rangle$ を初期値として新たに生成した整数の軌道
 $\equiv Ck_{i-j}$ を圧縮変換したところのビットストリーム
 である。

【図3】

カオス的鍵と鍵ストリームの生成・更新のフロー

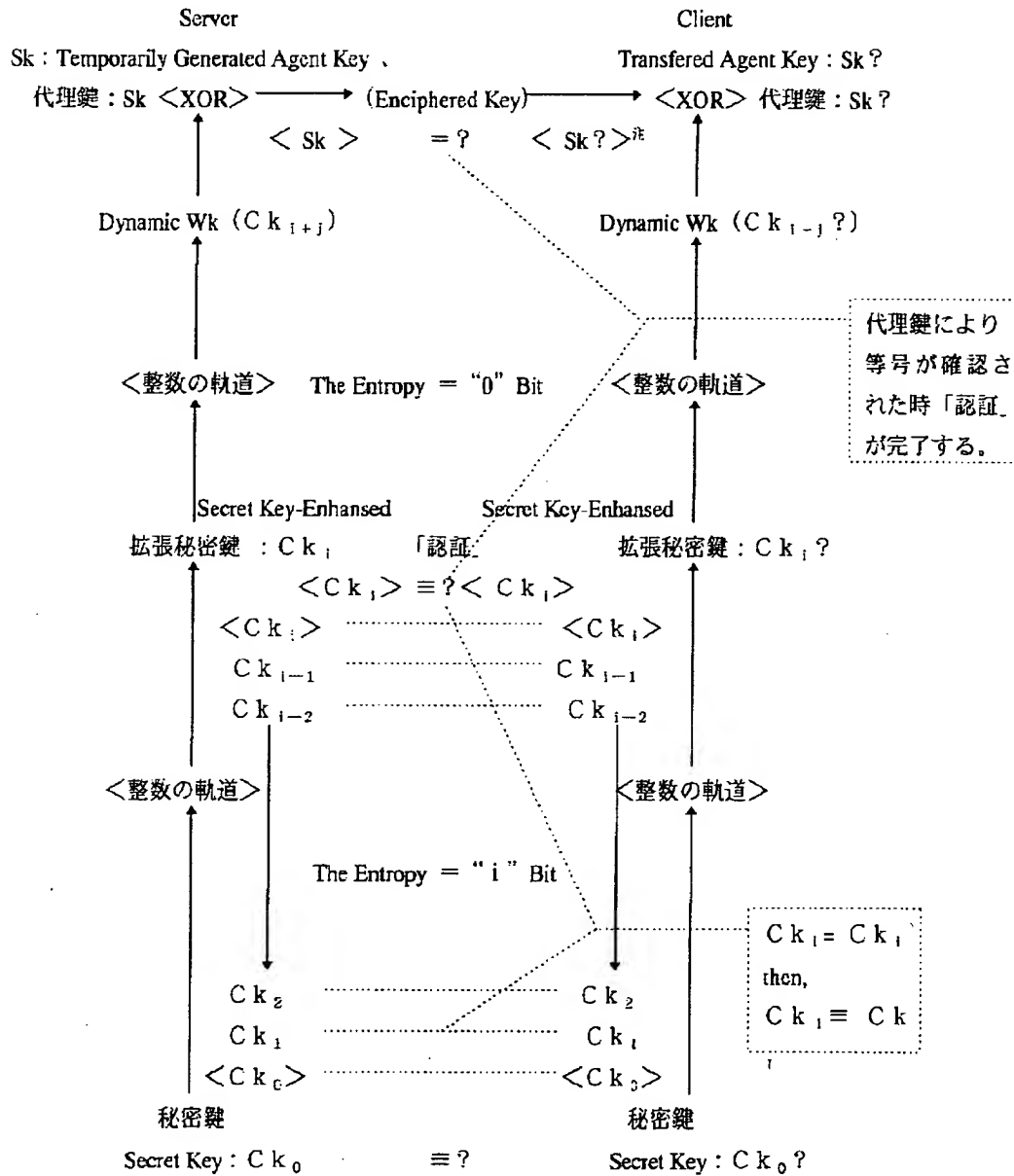


【図 4】

代理鍵の安全同期配送プロトコルと代理鍵による安全同期認証プロトコル：

< Secure Synchro-Transfer Protocol For Agent Key and

Secure Synchro-Authentication Protocol by Agent Key >

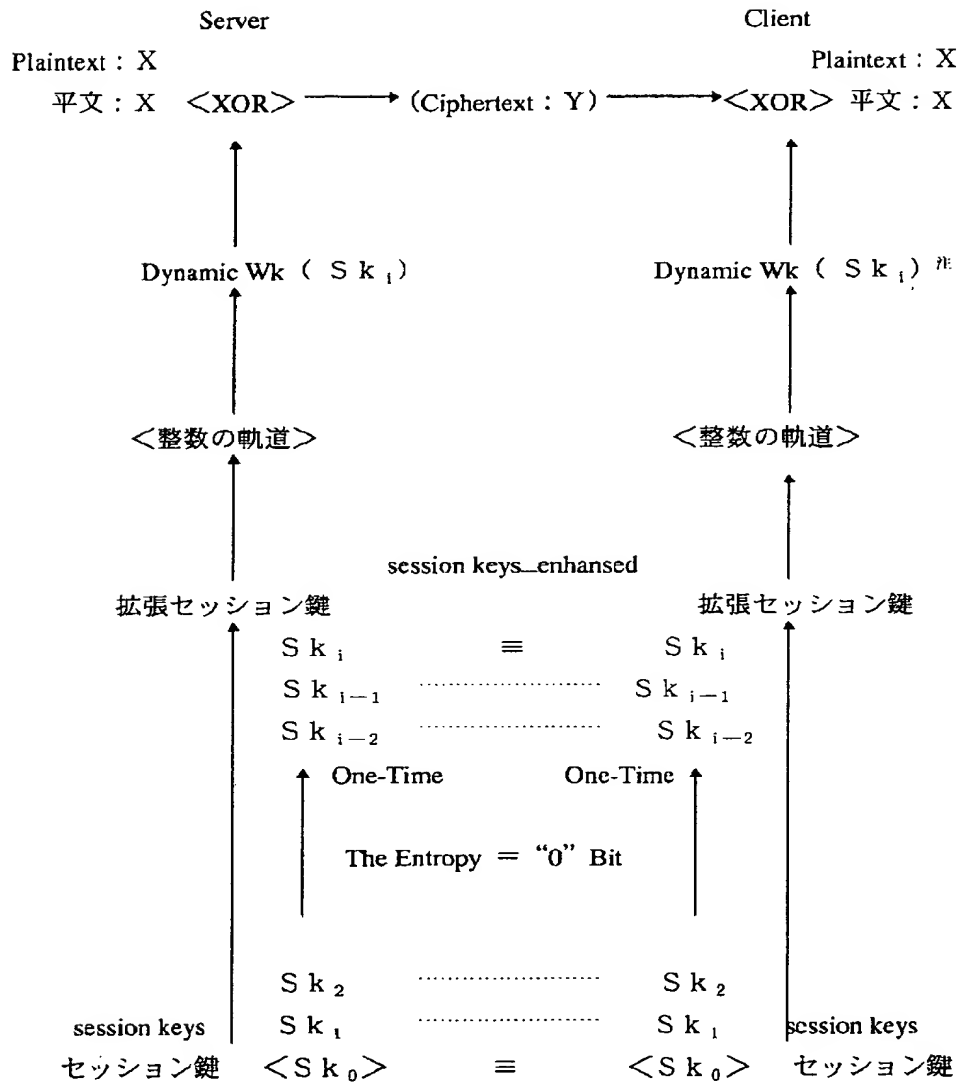


注)

$\langle \text{Sk. serv} \rangle = \langle \text{Sk. clin} \rangle$, then $\langle \text{Ck}_{i, \text{serv}} \rangle \equiv \langle \text{Ck}_{i, \text{clin}} \rangle$
 $\langle \text{Ck}_{i, \text{serv}} \rangle \equiv \langle \text{Ck}_{i, \text{clin}} \rangle$

【図5】

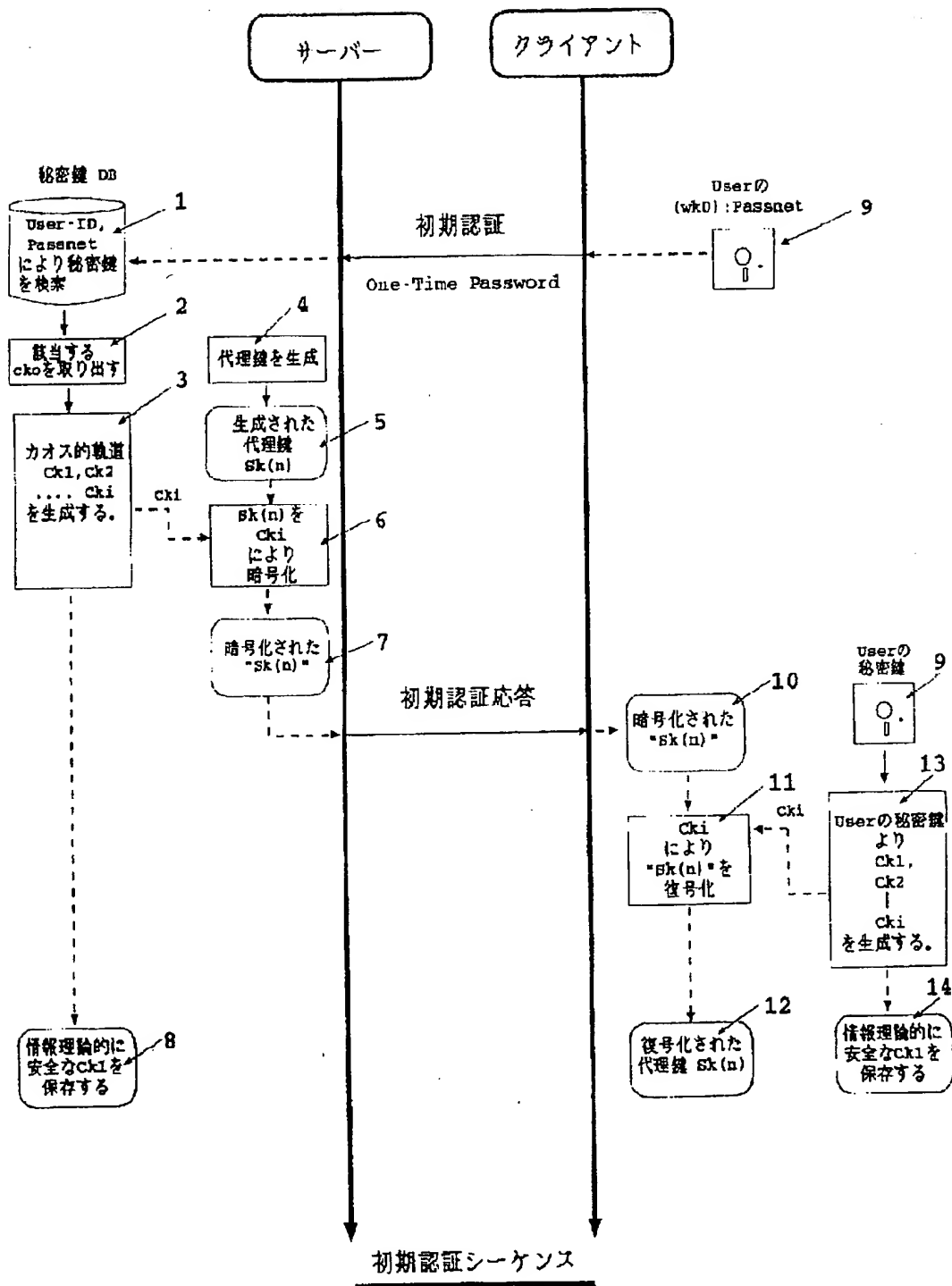
“One-Time” のセッション鍵群
 < one-time session keys >



注)

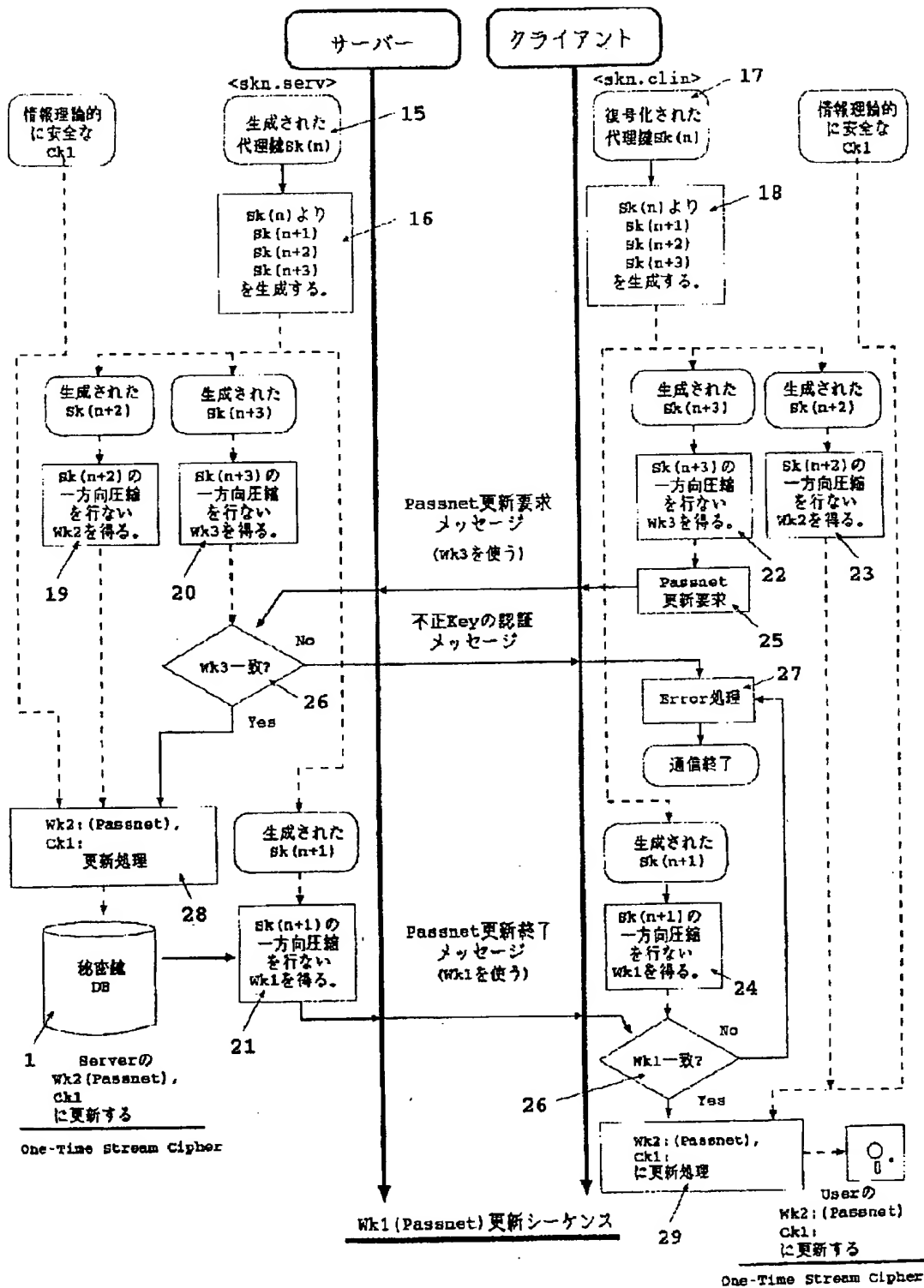
Dynamic Wk (Sk_i) : ($Sk_i, i=1$)、($Sk_i, i=2$)、($Sk_i, i=1$)
 をそれぞれ初期値としたところの“i”本のビットストリーム
 を指す：

【図6】

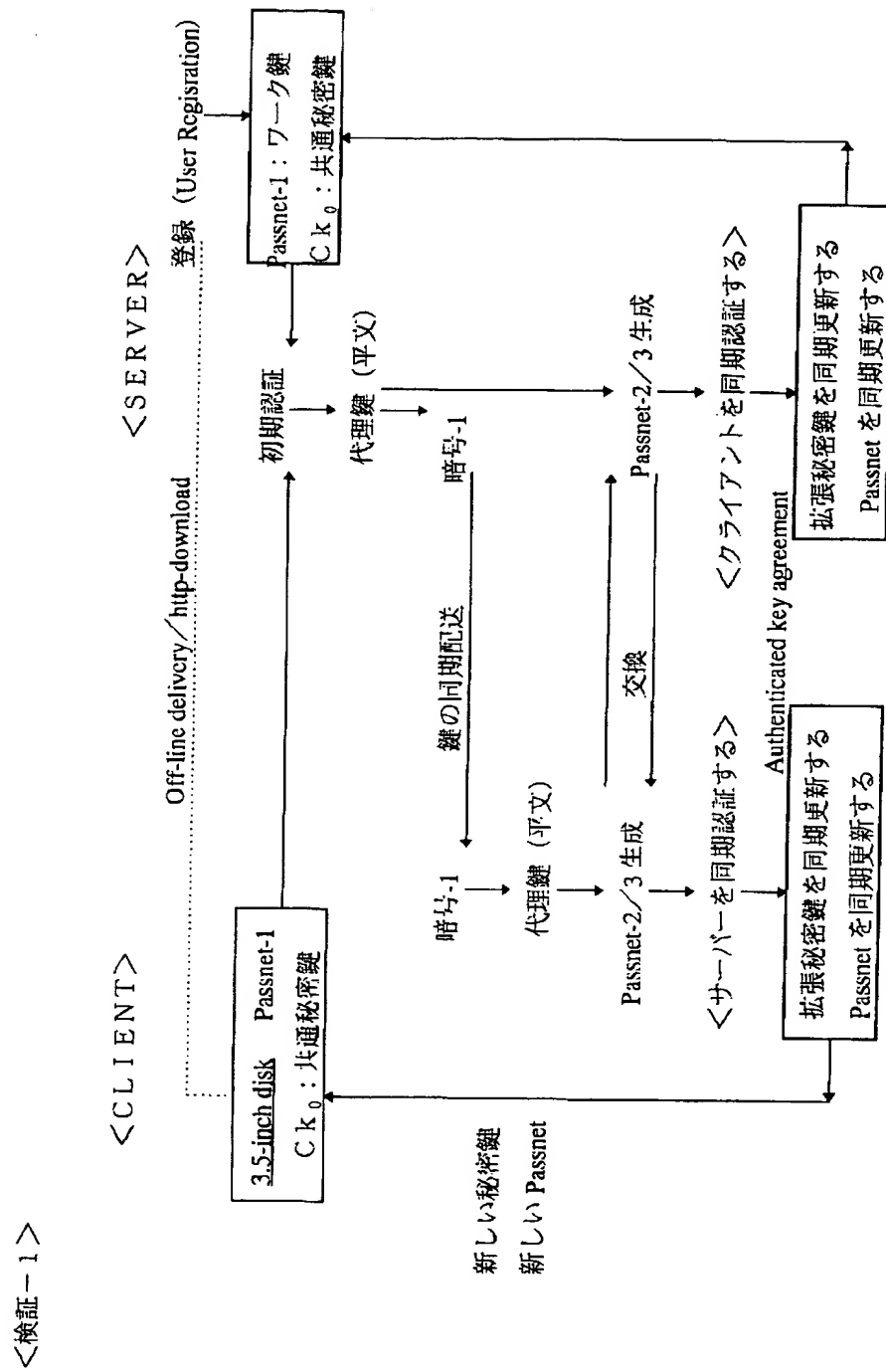


cki: 拡張秘密鍵

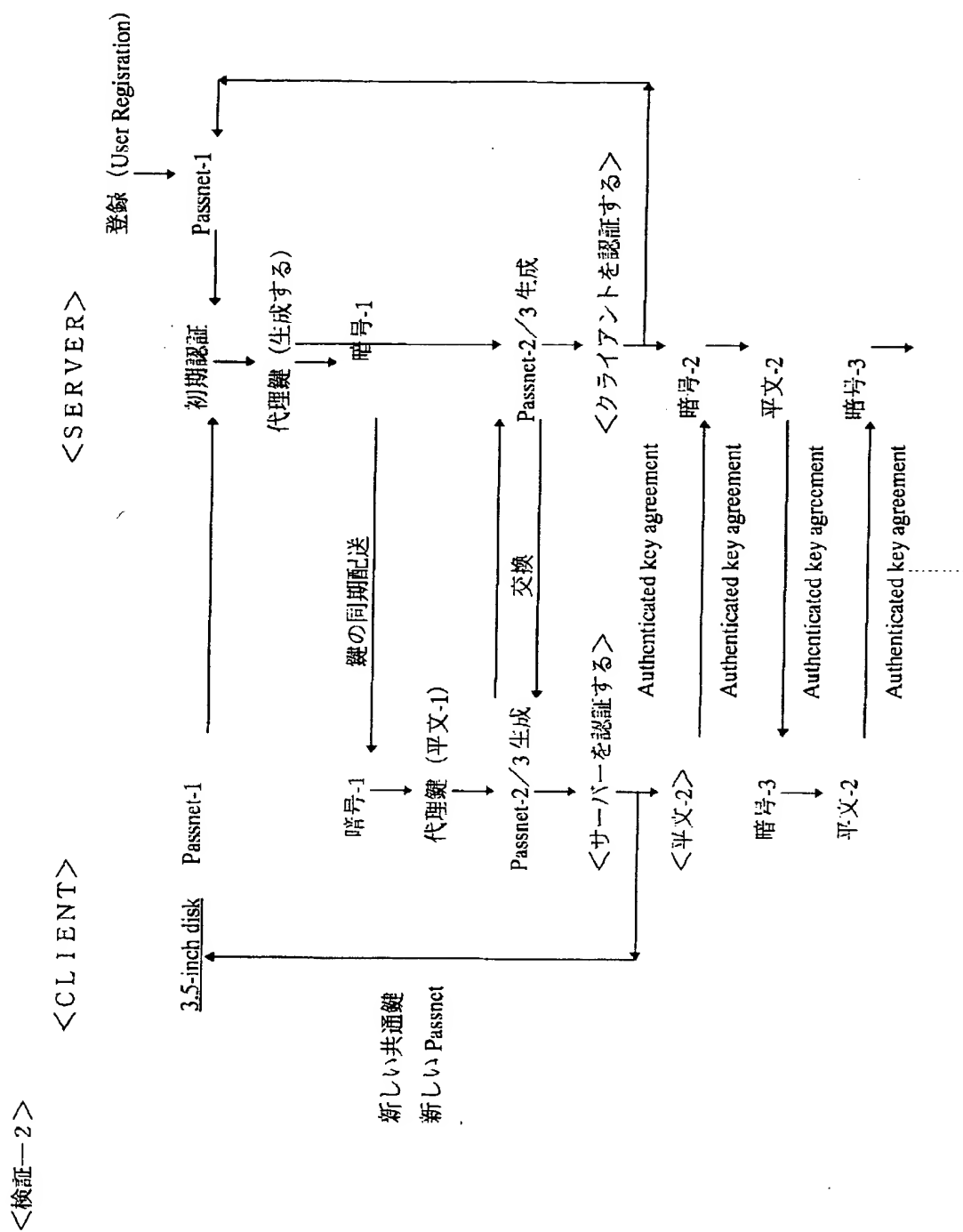
【図7】



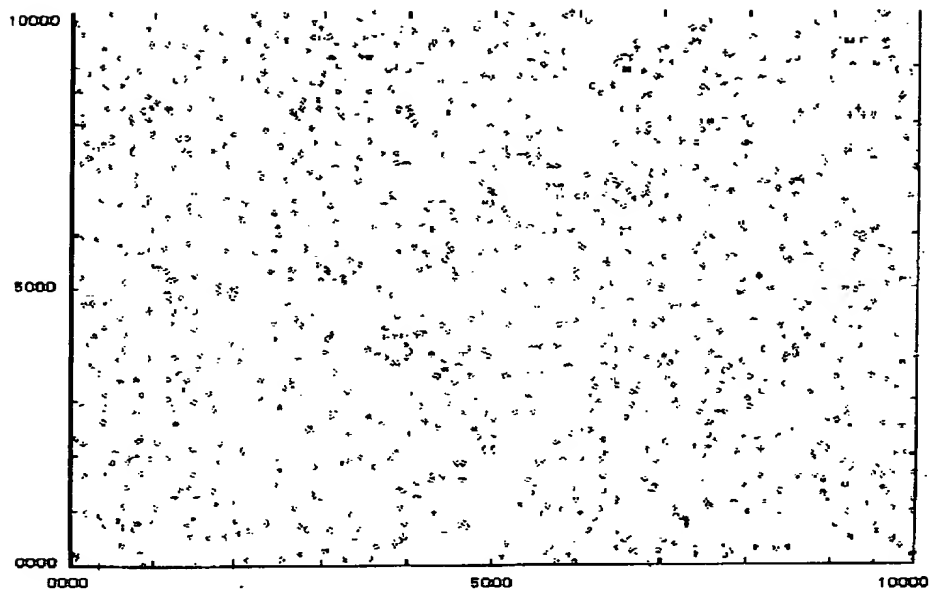
【図8】



【图 9】



【図11】



フロントページの続き

- (54) 【発明の名称】 完全守秘性暗号系を構成する鍵生成システム、認証付き鍵共有プロトコル、“One-Time Stream Cipher”、“One-Time passworded”及び鍵管理アルゴリズム